NEWOB

New Object Command: Creates a new copy of the specified object.

Level 1	→	Level 1
obj	→	*obj*

Keyboard Access: ◀ MEMORY NEWOB

Affected by Flags: Last Arguments (−55)

In order for NEWOB to immediately release the memory occupied by the original copy, flag −55 must be set so that the copy is not saved as a last argument.

Remarks: NEWOB has two main uses:

- NEWOB enables the purging of a library or backup object that has been recalled from a port. NEWOB creates a new, separate copy of the object in memory, thereby allowing the original copy to be purged.
- Creating a new copy of an object that originated in a larger composite object (such as a list) allows you to recover the memory associated with the larger object when that larger object is no longer needed.

Examples: `:0:BKUP1 RCL NEWOB :0:BKUP1 PURGE` recalls and purges the backup object *BKUP1*.

`3 GET NEWOB` retrieves the third element out of a list in the stack, recovering the memory occupied by the whole list.

Related Commands: MEM, PURGE

NEXT

NEXT Command: Ends definite loop structures.

See the FOR and START command entries for syntax information.

Keyboard Access:

PRG BRCH SHIFT BRCH

PRG BRCH FOR NEXT

Remarks: See the FOR and START keyword entries for more information.

Related Commands: FOR, START, STEP

NEXT

Next Operation: Returns but does not execute the next one or two steps of a program.

Keyboard Access: PRG NXT RUN NEXT

Affected by Flags: None

Related Commands: SST, SST↓

NOT

NOT Command: Returns the one's complement or logical inverse of the argument.

{}

Level 1	→	Level 1
#n_1	→	#n_2
T/F	→	0/1
"*string*$_1$"	→	"*string*$_2$"
'*symb*'	→	'NOT *symb*'

Keyboard Access:

(PRG) TEST (NXT) NOT

(MTH) BASE (NXT) LOGIC NOT

Affected by Flags: Numerical Results (−3), Binary Integer Wordsize (−5 through −10)

Remarks: When the argument is a binary integer or string, NOT complements each bit in the argument to produce the result.

- A binary integer is treated as a sequence of bits as long as the current wordsize.
- A string is treated as a sequence of bits, using 8 bits per character (that is, using the binary version of the character code).

When the argument is a real number or symbolic, NOT does a true/false test. The result is 1 (true) if the argument is zero; it is 0 (false) if the argument is nonzero. This test is usually done on a test result (T/F).

If the argument is an algebraic object, then the result is an algebraic of the form 'NOT *symb*' Execute →NUM (or set flag −3 before executing NOT) to produce a numeric result from the algebraic result.

Related Commands: AND, OR, XOR

NOVAL

INFORM Place Holder/Result Command: Place holder for reset and initial values in user-defined dialog boxes. NOVAL is returned to the stack when a field is empty.

Keyboard Access: (PRG) (NXT) IN NOVAL

Affected by Flags: None

Remarks: NOVAL is used to mark an empty field in a user-defined dialog box created with the INFORM command. INFORM defines fields sequentially. If default values are used for those fields, the defaults must be defined in the same order as the fields were defined. To skip over (not provide defaults for) some of the fields, use the NOVAL command.

After INFORM terminates, NOVAL is returned to the stack (on level 2) if a field is empty and OK or (ENTER) is selected.

Related Commands: INFORM

NSUB

Number of Sublist Command: Provides a way to access the current sublist position during an iteration of a program or command applied using DOSUBS.

Keyboard Access: (PRG) LIST PROC NSUB

Affected by Flags: None

Remarks: Returns an Undefined Local Name error if executed when DOSUBS is not active.

Related Commands: DOSUBS, ENDSUB

NUM

Character Number Command: Returns the character code *n* for the first character in the string.

{}

Level 1	→	Level 1
"string"	→	*n*

Keyboard Access:

◀ CHARS NUM

PRG TYPE NXT NUM

Affected by Flags: None

Remarks: The character codes are an extension of ISO 8859/1. Codes 128 through 159 are unique to the HP 48.

The following tables show the relation between character codes (results of NUM, arguments to CHR) and characters (results of CHR, arguments to NUM).

Character Codes (0 — 127)

NUM	CHR	NUM	CHR	NUM	CHR	NUM	CHR
0	▪	32		64	@	96	`
1	▪	33	!	65	A	97	a
2	▪	34	"	66	B	98	b
3	▪	35	#	67	C	99	c
4	▪	36	$	68	D	100	d
5	▪	37	%	69	E	101	e
6	▪	38	&	70	F	102	f
7	▪	39	'	71	G	103	g
8	▪	40	(	72	H	104	h
9	▪	41	)	73	I	105	i
10	▪	42	*	74	J	106	j
11	▪	43	+	75	K	107	k
12	▪	44	,	76	L	108	l
13	▪	45	-	77	M	109	m
14	▪	46	.	78	N	110	n
15	▪	47	/	79	O	111	o
16	▪	48	0	80	P	112	p
17	▪	49	1	81	Q	113	q
18	▪	50	2	82	R	114	r
19	▪	51	3	83	S	115	s
20	▪	52	4	84	T	116	t
21	▪	53	5	85	U	117	u
22	▪	54	6	86	V	118	v
23	▪	55	7	87	W	119	w
24	▪	56	8	88	X	120	x
25	▪	57	9	89	Y	121	y
26	▪	58	:	90	Z	122	z
27	▪	59	;	91	[	123	{
28	▪	60	<	92	\	124	|
29	▪	61	=	93	]	125	}
30	▪	62	>	94	^	126	~
31	▪	63	?	95	_	127	▪

Character Codes (128 — 255)

NUM	CHR	NUM	CHR	NUM	CHR	NUM	CHR
128	◢	160		192	À	224	à
129	◥	161	¡	193	Á	225	á
130	▽	162	¢	194	Â	226	â
131	√	163	£	195	Ã	227	ã
132	∫	164	¤	196	Ä	228	ä
133	Σ	165	¥	197	Å	229	å
134	▶	166	¦	198	Æ	230	æ
135	π	167	§	199	Ç	231	ç
136	∂	168	¨	200	È	232	è
137	≤	169	©	201	É	233	é
138	≥	170	ª	202	Ê	234	ê
139	≠	171	«	203	Ë	235	ë
140	α	172	¬	204	Ì	236	ì
141	→	173	-	205	Í	237	í
142	←	174	®	206	Î	238	î
143	↓	175	¯	207	Ï	239	ï
144	↑	176	°	208	Þ	240	ð
145	γ	177	±	209	Ñ	241	ñ
146	δ	178	²	210	Ò	242	ò
147	ε	179	³	211	Ó	243	ó
148	η	180	´	212	Ô	244	ô
149	θ	181	µ	213	Õ	245	õ
150	λ	182	¶	214	Ö	246	ö
151	ρ	183	·	215	×	247	÷
152	σ	184	¸	216	Ø	248	ø
153	τ	185	¹	217	Ù	249	ù
154	ω	186	º	218	Ú	250	ú
155	Δ	187	»	219	Û	251	û
156	Π	188	¼	220	Ü	252	ü
157	Ω	189	½	221	Ý	253	ý
158	■	190	¾	222	Þ	254	þ
159	∞	191	¿	223	ß	255	ÿ

Related Commands: CHR, POS, REPL, SIZE, SUB

→NUM

Evaluate to Number Command: Evaluates a symbolic argument object (other than a list) and returns the numerical result.

{}

Level 1	→	Level 1
obj_{symb}	→	z

Keyboard Access: ◀ →NUM

Affected by Flags: None

Remarks: →NUM repeatedly evaluates a symbolic argument until a numerical result is achieved. The effect is the same as evaluating the symbolic argument in Numerical Result mode (flag −3 set).

Related Commands: EVAL, SYSEVAL

NUMX

Number of X-Steps Command: Sets the number of x-steps for each y-step in 3D perspective plots.

{}

Level 1	→	Level 1
n_x	→	

Keyboard Access: ◀ PLOT NXT 3D VIEW NXT NUMX

Affected by Flags: None

Remarks: The number of x-steps is the number of independent variable points plotted for each dependent variable point plotted. This number must be 2 or more. This value is stored in the reserved

variable VPAR. YSLICE is the only 3D plot type that does not use this value.

Related Commands: NUMY

NUMY

Number of Y-Steps Command: Sets the number of y-steps across the view volume in 3D perspective plots.

{}

Level 1	→	Level 1
n_y	→	

Keyboard Access: ◀ PLOT NXT 3D VPAR NXT NUMY

Affected by Flags: None

Remarks: The number of y-steps is the number of dependent variable points plotted across the view volume. This number must be 2 or more. This value is stored in the reserved variable VPAR.

Related Commands: NUMX

NΣ

Number of Rows Command: Returns the number of rows in the current statistical matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	n_{rows}

Keyboard Access: ◀ STAT SUMS NΣ

Affected by Flags: None

Related Commands: ΣX, ΣX*Y, ΣX^2, ΣY, ΣY^2

OBJ→

Object to Stack Command: Separates an object into its components onto the stack. For some object types, the *number* of components is returned to level 1.

Level 1	→	Level n+1 ...	Level 2	Level 1
(x,y)	→		x	y
{ obj_1 ... obj_n }	→	obj_1	obj_n	n
[x_1 ... x_n]	→	x_1	x_n	{ n }
[[x_{11} ... x_{mn}]]	→	x_{11}	x_{mn}	{ m n }
"obj"	→			evaluated-object
'symb'	→	arg_1 ... arg_n	n	'function'
x_unit	→		x	1_unit
:tag:obj	→		obj	"tag"

Keyboard Access:

◀ CHARS NXT OBJ→

PRG TYPE OBJ→

Affected by Flags: None

Remarks: If the argument is a complex number, list, array, or string, OBJ→ provides the same functions as C→R, LIST→, ARRY→, and STR→, respectively. For lists, OBJ→ also returns the number of list elements. If the argument is an array, OBJ→ also returns the dimensions { m n } of the array, where m is the number of rows and n is the number of columns.

For algebraic objects, OBJ→ returns the arguments of the top-level (least-nested) function (arg_1 ... arg_n), the number of arguments

of the top-level function (n), and the name of the top-level function (*function*).

If the argument is a string, the object sequence defined by the string is executed.

Example: The command sequence `'∫ (0,1,SIN(X),X)' OBJ→` returns:

6:	0	first argument
5:	1	second argument
4:	'SIN(X)'	third argument
3:	'X'	fourth argument
2:	4	number of arguments for ∫
1:	∫	function name

Related Commands: ARRY→, C→R, DTAG, EQ→, LIST→, R→C, STR→, →TAG

OCT

Octal Mode Command: Selects octal base for binary integer operations. (The default base is decimal.)

Keyboard Access: MTH BASE OCT

Affected by Flags: Binary Integer Wordsize (−5 through −10), Binary Integer Base (−11, −12)

Remarks: Binary integers require the prefix #. Binary integers entered and returned in octal base automatically show the suffix o. If the current base is not octal, enter an octal number by ending it with o. It will be displayed in the current base when entered.

The current base does not affect the internal representation of binary integers as unsigned binary numbers.

Related Commands: BIN, DEC, HEX, RCWS, STWS

OFF

Off Command: Turns off the calculator.

Keyboard Access: PRG NXT PUT NXT OFF

Affected by Flags: None

Remarks: When executed from a program, that program will resume execution when the calculator is turned on. This provides a programmable "autostart."

Related Commands: CONT, HALT, KILL

OLDPRT

Old Printer Command: Modifies the remapping string in the reserved variable *PRTPAR* so that the extended character set of the HP 48 matches that of the HP 82240A Infrared Printer.

Keyboard Access: ← I/O PRINT PRTPR OLDPR

Affected by Flags: None

Remarks: The character set in the HP 82240A Infrared Printer does not match the HP 48 character set:

- 24 characters in the HP 48 character set are not available in the HP 82240A Infrared Printer. (From the table in the keyword listing for NUM, these characters are numbers 129, 130, 143-157, 159, 166, 169, 172, 174, 184, and 185.) The HP 82240A prints a ▒ in substitution.
- Many characters in the extended character table (character codes 128 through 255) do not have the same character code. For example, the « character has code 171 in the HP 48 and code 146 in the HP 82240A Infrared Printer.

To use the CHR command to print extended characters with an HP 82240A Infrared Printer, first execute OLDPRT. The remapping string modified by OLDPRT is the second parameter in *PRTPAR*. This string (which is empty in the default state) changes the character code of each byte to match the codes in the HP 82240A Infrared Printer character table.

OLDPRT

To cancel OLDPRT character mapping, purge the variable *PRTPAR*, or enter `0 PRTPAR PUT`.

To print a string containing graphics data, disable OLDPRT.

Related Commands: CR, DELAY, PRLCD, PRST, PRSTC, PRVAR, PR1

OPENIO

Open I/O Port Command: Opens the serial port or the IR port using the I/O parameters in the reserved variable *IOPAR*.

Keyboard Access: ◀ I/O NXT SERIAL OPENIO

Affected by Flags: I/O Device (−33)

Remarks: Since all HP 48 Kermit-protocol commands automatically effect an OPENIO first, OPENIO is not normally needed, but can be used if an I/O transmission does not work. OPENIO is necessary for interaction with devices that interpret a closed port as a break.

OPENIO is also necessary for the automatic reception of data into the input buffer using non-Kermit commands. If the port is closed, incoming characters are ignored. If the port is open, incoming characters are automatically placed in the input buffer. These characters can be detected with BUFLEN, and can be read out of the input buffer using SRECV.

If the port is already open, OPENIO does not affect the data in the input buffer. However, if the port is closed, executing OPENIO clears the data in the input buffer.

For more information, refer to the reserved variable *IOPAR* in appendix D, "Reserved Variables."

Related Commands: BUFLEN, CLOSEIO, SBRK, SRECV, STIME, XMIT

OR

OR Function: Returns the logical OR of two arguments.

{}

Level 2	Level 1	→	Level 1
$\#n_1$	$\#n_2$	→	$\#n_3$
"$string_1$"	"$string_2$"	→	"$string_3$"
T/F_1	T/F_2	→	0/1
T/F	'$symb$'	→	'T/F OR $symb$'
'$symb$'	T/F	→	'$symb$ OR T/F'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ OR $symb_2$'

Keyboard Access:

MTH BASE NXT LOGIC OR

PRG TEST NXT OR

Affected by Flags: Numerical Results (−3), Binary Integer Wordsize (−5 through −10)

Remarks: When the arguments are binary integers or strings, OR does a bit-by-bit (base 2) logical comparison.

- An argument that is a binary integer is treated as a sequence of bits as long as the current wordsize. Each bit in the result is determined by comparing the corresponding bits (bit_1 and bit_2) in the two arguments as shown in the following table.

bit_1	bit_2	bit_1 **OR** bit_2
0	0	0
0	1	1
1	0	1
1	1	1

- An argument that is a string is treated as a sequence of bits, using 8 bits per character (that is, using the binary version of the character code). The two string arguments must be the same length.

OR

When the arguments are real numbers or symbolics, OR simply does a true/false test. The result is 1 (true) if either or both arguments are nonzero; it is 0 (false) if both arguments are zero. This test is usually done to compare two test results.

If either or both of the arguments are algebraic objects, then the result is an algebraic of the form '$symb_1$ OR $symb_2$'. Execute →NUM (or set flag −3 before executing OR) to produce a numeric result from the algebraic result.

Related Commands: AND, NOT, XOR

ORDER

Order Variables Command: Reorders the variables in the current directory (shown in the VAR menu) to the order specified.

Level 1	→	Level 1
{ $global_1$ ... $global_n$ }	→	

Keyboard Access: ◀ MEMORY NXT ORDER

Affected by Flags: None

Remarks: The names that appear first in the list will be the first to appear in the VAR menu. Variables not specified in the list are placed after the reordered variables.

If the list includes the name of a large subdirectory, there may be insufficient memory to execute ORDER.

Related Commands: VARS

OVER

Over Command: Returns a copy to stack level 1 of the object in level 2.

Level 2	Level 1	→	Level 3	Level 2	Level 1
obj_1	obj_2	→	obj_1	obj_2	obj_1

Keyboard Access: ◀ STACK OVER

Affected by Flags: None

Related Commands: PICK, ROLL, ROLLD, ROT, SWAP

PARAMETRIC

Parametric Plot Type Command: Sets the plot type to PARAMETRIC.

Keyboard Access: ◀ PLOT PTYPE PARA

Affected by Flags: Simultaneous Plotting (−28), Curve Filling (−28)

Remarks: When the plot type is PARAMETRIC, the DRAW command plots the current equation as a complex-valued function of one real variable. The current equation is specified in the reserved variable *EQ*. The plotting parameters are specified in the reserved variable *PPAR*, which has the following form:

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

For plot type PARAMETRIC, the elements of *PPAR* are used as follows:

- (x_{min}, y_{min}) is a complex number specifying the lower left corner of *PICT* (the lower left corner of the display range). The default value is (−6.5,−3.1).

- (x_{max}, y_{max}) is a complex number specifying the upper right corner of *PICT* (the upper right corner of the display range). The default value is (6.5,3.2).
- *indep* is a list containing a name that specifies the independent variable, and two numbers specifying the minimum and maximum values for the independent variable (the plotting range). Note that the default value is *X*. If *X* is not modified and included in a list with a plotting range, the values in (x_{min}, y_{min}) and (x_{max}, y_{max}) are used as the plotting range, which generally leads to meaningless results.
- *res* is a real number specifying the interval, in user-unit coordinates, between values of the independent variable. The default value is 0, which specifies an interval equal to 1/130 of the difference between the maximum and minimum values in *indep* (the plotting range).
- *axes* is a list containing one or more of the following, in the order listed: a complex number specifying the user-unit coordinates of the plot origin, a list specifying the tick-mark annotation, and two strings specifying labels for the horizontal and vertical axes. The default value is (0,0).
- *ptype* is a command name specifying the plot type. Executing the command PARAMETRIC places the name PARAMETRIC in *PPAR*.
- *depend* is a name specifying a label for the vertical axis. The default value is *Y*

The contents of *EQ* must be an expression or program; it cannot be an equation. It is evaluated for each value of the independent variable. The results, which must be complex numbers, give the coordinates of the points to be plotted. Lines are drawn between plotted points unless flag −31 is set.

If flag −28 is set, all equations are plotted simultaneously.

See chapter 23 of the *HP 48 User's Guide* for an example using the PARAMETRIC plot type.

Related Commands: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

PARITY

Parity Command: Sets the parity value in the reserved variable *IOPAR*.

{}

Level 1	→	Level 1
n_{parity}	→	

Keyboard Access: ◀ I/O PARITY

Affected by Flags: None

Remarks: Legal values are shown below. A negative value means the HP 48 does not check parity on bytes received during Kermit transfers or with SRECV. Parity is still used during data transmission, however.

n-Value	Meaning
0	no parity (the default value)
1	odd parity
2	even parity
3	mark
4	space

For more information, refer to the reserved variable *IOPAR* (*I/O parameters*) in appendix D, "Reserved Variables."

Related Commands: BAUD, CKSM, TRANSIO

PARSURFACE

PARSURFACE Plot Type Command: Sets plot type to PARSURFACE.

Keyboard Access: ◀ PLOT NXT PTYPE PARSU

Affected by Flags: None

Remarks: When plot type is set to PARSURFACE, the DRAW command plots an image graph of a 3-vector-valued function of two variables. PARSURFACE requires values in the reserved variables *EQ*, *VPAR*, and *PPAR*.

VPAR is made up of the following elements:

$\{\ x_{left}\ x_{right}\ y_{near}\ y_{far}\ z_{low}\ z_{high}\ x_{min}\ x_{max}\ y_{min}\ y_{max}\ x_{eye}\ y_{eye}\ z_{eye}\ x_{step}\ y_{step}\ \}$

For plot type PARSURFACE, the elements of *VPAR* are used as follows:

- x_{left} and x_{right} are real numbers that specify the width of the view space.
- y_{near} and y_{far} are real numbers that specify the depth of the view space.
- z_{low} and z_{high} are real numbers that specify the height of the view space.
- x_{min} and x_{max} are real numbers the specify the input region's width. The default value is $(-1, 1)$.
- y_{min} and y_{max} are real numbers that specify the input region's depth. The default value is $(-1, 1)$.
- x_{eye}, y_{eye}, and z_{eye} are real numbers that specify the point in space from which the graph is viewed.
- x_{step} and y_{step} are real numbers that set the number of x-coordinates versus the number of y-coordinates plotted.

The plotting parameters are specified in the reserved variable *PPAR*, which has this form:

$\{\ (x_{min}, y_{min})\ (x_{max}, y_{max})\ indep\ res\ axes\ ptype\ depend\ \}$

For plot type PARSURFACE, the elements of *PPAR* are used as follows:

- (x_{min}, y_{min}) is not used.
- (x_{max}, y_{max}) is not used.
- *indep* is a name specifying the independent variable. The default value of *indep* is *X*.
- *res* is not used.
- *axes* is not used.
- *ptype* is a command name specifying the plot type. Executing the command PARSURFACE places the name PARSURFACE in *ptype*.
- *depend* is a name specifying the dependent variable. The default value is *Y*

Related Commands: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

PATH

Current Path Command: Returns a list specifying the path to the current directory.

Level 1 →	Level 1
→	{ HOME *directory-name*$_1$ ... *directory-name*$_n$ }

Keyboard Access: ◀ MEMORY DIR PATH

Affected by Flags: None

Remarks: The first directory is always *HOME*, and the last directory is always the current directory.

If a program needs to switch to a specific directory, it can do so by evaluating a directory list, such as one created earlier by PATH.

Related Commands: CRDIR, HOME, PGDIR, UPDIR

PCOEF

Monic Polynomial Coefficients Command: Returns the coefficients of a monic polynomial (a polynomial with a leading coefficient of 1) having specific roots.

{}

Level 1	→	Level 1
[*array*]$_{roots}$	→	[*array*]$_{coefficients}$

Keyboard Access: ◀ SOLVE POLY PCOEF

Affected by Flags: None

Remarks: The argument must be a real or complex array of length n containing the polynomial's roots. The result is a real or complex vector of length $n+1$ containing the coefficients listed from highest order to lowest, with a leading coefficient of 1.

Example: Find the polynomial that has the roots 2, −3, 4, −5:

`[ 2 -3 4 -5 ] PCOEF` returns `[ 1 2 -25 -26 120 ]`, representing the polynomial $x^4 + 2x^3 - 25x^2 - 26x + 120$.

Related Commands: PEVAL, PROOT

PCONTOUR

PCONTOUR Plot Type Command: Sets the plot type to PCONTOUR.

Keyboard Access: ◀ PLOT NXT 2D PTYPE PCON

Affected by Flags: None

Remarks: When plot type is set PCONTOUR, the DRAW command plots a contour-map view of a scalar function of two variables. PCONTOUR requires values in the reserved variables *EQ*, *VPAR*, and *PPAR*.

VPAR is made up of the following elements:

$\langle x_{left}\ x_{right}\ y_{near}\ y_{far}\ z_{low}\ z_{high}\ x_{min}\ x_{max}\ y_{min}\ y_{max}\ x_{eye}\ y_{eye}\ z_{eye}\ x_{step}\ y_{step} \rangle$.

For plot type PCONTOUR, the elements of *VPAR* are used as follows:

- x_{left} and x_{right} are real numbers that specify the width of the view space.
- y_{near} and y_{far} are real numbers that specify the depth of the view space.
- z_{low} and z_{high} are real numbers that specify the height of the view space.
- x_{min} and x_{max} are not used.
- y_{min} and y_{max} are not used.
- x_{eye}, y_{eye}, and z_{eye} are real numbers that specify the point in space from which the graph is viewed.
- x_{step} and y_{step} are real numbers that set the number of x-coordinates versus the number of y-coordinates plotted.

The plotting parameters are specified in the reserved variable *PPAR*, which has this form:

$\langle (x_{min}, y_{min})\ (x_{max}, y_{max})\ indep\ res\ axes\ ptype\ depend \rangle$

For plot type PCONTOUR, the elements of *PPAR* are used as follows:

- (x_{min}, y_{min}) is not used.
- (x_{max}, y_{max}) is not used.

- *indep* is a name specifying the independent variable. The default value of *indep* is X.
- *res* is not used.
- *axes* is not used.
- *ptype* is a command name specifying the plot type. Executing the command PCONTOUR places the name PCONTOUR in *ptype*.
- *depend* is a name specifying the dependent variable. The default value is Y.

Related Commands: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

PCOV

Population Covariance Command: Returns the population covariance of the independent and dependent data columns in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	$x_{\text{pcovariance}}$

Keyboard Access: ◀ STAT [illegible] NXT [illegible]

Affected by Flags: None

Remarks: The columns are specified by the first two elements in reserved variable ΣPAR, set by XCOL and YCOL respectively. If ΣPAR does not exist, PCOV creates it and sets the elements to their default values (1 and 2).

The population covariance is calculated with the following formula:

$$\frac{1}{n}\sum_{k=1}^{n}(x_{kn_1}-\overline{x_{n_1}})(x_{kn_2}-\overline{x_{n_2}})$$

where x_{kn_1} is the kth coordinate value in column n_1, x_{kn_2} is the kth coordinate value in the column n_2, $\overline{x_{n_1}}$ is the mean of the data in column n_1, $\overline{x_{n_2}}$ is the mean of the data in column n_2, and n is the number of data points.

Related Commands: COLΣ, CORR, COV, PREDX, PREDY, XCOL, YCOL

PDIM

PICT-Dimension Command: Replaces *PICT* with a blank *PICT* of the specified dimensions.

{ }

Level 2	Level 1	→	Level 1
$(x_{\text{min}}, y_{\text{min}})$	$(x_{\text{max}}, y_{\text{max}})$	→	
$\#n_{\text{width}}$	$\#m_{\text{height}}$	→	

Keyboard Access: PRG [illegible] PDIM

Affected by Flags: None

Remarks: If the arguments are complex numbers, PDIM changes the size of *PICT* and makes the arguments the new values of $(x_{\text{min}}, y_{\text{min}})$ and $(x_{\text{max}}, y_{\text{max}})$ in the reserved variable *PPAR*. Thus, the scale of a subsequent plot is not changed. If the arguments are binary integers, *PPAR* remains unchanged, so the scale of a subsequent plot *is* changed.

PICT cannot be smaller than 131 pixels wide × 64 pixels high, nor wider than 2048 pixels (height is unlimited).

Related Commands: PMAX, PMIN

PERM

Permutations Function: Returns the number of possible permutations of n items taken m at a time.

{}

Level 2	Level 1	→	Level 1
n	m	→	$P_{n,m}$
'$symb_n$'	m	→	'PERM($symb_{n,m}$)'
n	'$symb_m$'	→	'PERM(n, $symb_m$)'
'$symb_n$'	'$symb_m$'	→	'PERM($symb_n$, $symb_m$)'

Keyboard Access: MTH NXT PROB PERM

Affected by Flags: Numerical Results (−3)

Remarks: The formula used to calculate $P_{n,m}$ is this:

$$P_{n,m} = \frac{n!}{(n-m)!}$$

The arguments n and m must each be less than 10^{12}.

Related Commands: COMB, !

PEVAL

Polynomial Evaluation Command: Evaluates an n-degree polynomial at x.

{}

Level 2	Level 1	→	Level 1
[*array*]$_{coefficients}$	x	→	$p(x)$

Keyboard Access: ↶ SOLVE POLY PEVAL

Affected by Flags: None

Remarks: The arguments must be an array of length $n+1$ containing the polynomial's coefficients listed from highest order to lowest, and the value x at which the polynomial is to be evaluated.

Example: Evaluate the polynomial $x^4 + 2x^3 - 25x^2 - 26x + 120$ at $x = 8$:

`[ 1 2 -25 -26 120 ] 8` returns `3432`.

Related Commands: PCOEF, PROOT

PGDIR

Purge Directory Command: Purges the named directory (whether empty or not).

{}

Level 1	→	Level 1
'*global*'	→	

Keyboard Access: ↶ MEMORY DIR PGDIR

Affected by Flags: None

Related Commands: CLVAR, CRDIR, HOME, PATH, PURGE, UPDIR

PICK

Pick Object Command: Copies the contents of a specified level to level 1.

Level n+1..	Level 2	Level 1	→	Level n+1..	Level 2	Level 1
obj_n ..	obj_1	n	→	obj_n ..	obj_1	obj_n

Keyboard Access: ◀ STACK PICK

Affected by Flags: None

Related Commands: DUP, DUPN, DUP2, OVER, ROLL, ROLLD, ROT, SWAP

PICT

PICT Command: Puts the name PICT on the stack.

Level 1	→	Level 1
	→	*PICT*

Keyboard Access: PRG PICT PICT

Affected by Flags: None

Remarks: *PICT* is the name of a storage location in calculator memory containing the current graphics object. The command PICT enables access to the contents of that memory location as if it were a variable. Note, however, that *PICT* is *not* a variable as defined in the HP 48: its name cannot be quoted, and only graphics objects may be "stored" in it.

If a graphics object smaller than 131 wide × 64 pixels high is stored in *PICT*, it is enlarged to 131 × 64. A graphics object of unlimited pixel height and up to 2048 pixels wide can be stored in *PICT*.

Examples: PICT RCL returns the current graphics object to the stack.

GRAPHIC 131 × 64 PICT STO stores a graphics object in *PICT*, making it the current graphics object.

Related Commands: GOR, GXOR, NEG, PICTURE, PVIEW, RCL, REPL, SIZE, STO, SUB

PICTURE

Picture Environment Command: Selects the Picture environment (selects the graphics display and activates the graphics cursor and Picture menu).

Keyboard Access: ◀ PICTURE

Affected by Flags: None

Remarks: When executed from a program, PICTURE suspends program execution until CANCEL is pressed.

Example: This program:

```
« "Press CANCEL to return↵to stack" 1 DISP
3 WAIT PICTURE »
```

displays a message for 3 seconds, then selects the Picture environment. (The ↵ character in the program indicates a linefeed.)

Related Commands: PICTURE, PVIEW, TEXT

PINIT

Port Initialize Command: Initializes all currently active ports. Does not affect data already stored in a port.

Keyboard Access: ◀ LIBRARY NXT PINIT

Affected by Flags: None

Related Commands: None

Remarks: PINIT is particularly useful when using a plug-in card that can hold multiple ports. It stores and then purges an object in each port (128K partition) that can be accessed at the time the command is executed. This has the effect of initializing each port without disturbing any previously-stored data.

PIXOFF

Pixel Off Command: Turns off the pixel at the specified coordinate in *PICT*.

Level 1	→	Level 1
(*x*, *y*)	→	
{ #*n* #*m* }	→	

Keyboard Access: PRG PICT NXT PIXOFF

Affected by Flags: None

Related Commands: PIXON, PIX?

PIXON

Pixel On Command: Turns on the pixel at the specified coordinate in *PICT*.

Level 1	→	Level 1
(*x*, *y*)	→	
{ #*n* #*m* }	→	

Keyboard Access: PRG PICT NXT PIXON

Affected by Flags: None

Related Commands: PIXOFF, PIX?

PIX?

Pixel On? Command: Tests whether the specified pixel in *PICT* is on; returns 1 (true) if the pixel is on, and 0 (false) if the pixel is off.

Level 1	→	Level 1
(*x*, *y*)	→	0/1
{ #*n* #*m* }	→	0/1

Keyboard Access: PRG PICT NXT PIX?

Affected by Flags: None

Related Commands: PIXON, PIXOFF

PKT

Packet Command: Used to send command "packets" (and receive requested data) to a Kermit server.

{}

Level 2	Level 1	→	Level 1
"*data*"	"*type*"	→	"*response*"

Keyboard Access: ◀ I/O SERVER PKT

Affected by Flags: I/O Device (−33), I/O Messages (−39)

The I/O Data Format flag (−35) can be significant if the server sends back more than one packet.

Remarks: To send HP 48 objects, use SEND.

PKT allows additional commands to be sent to a Kermit server. For more information, refer to *Using MS-DOS Kermit* by Christine M. Gianone, Digital Press, 1990; or *KERMIT, A File Transfer Protocol* by Frank da Cruz, Digital Press, 1987, especially chapter 11, "The Client/Server Model."

The packet data, packet type, and the response to the packet transmission are all in string form. PKT first does an I (*initialization*) packet exchange with the Kermit server, then sends the server a packet constructed from the data and packet-type arguments supplied to PKT. The response to PKT will be either an acknowledging string (possibly blank) or an error packet (see KERRM).

For the *type* argument, only the first letter is significant.

Examples: A PKT command to send a *g*eneric *d*irectory request is "D" "G" PKT.

To send a *host command* packet, use a command from the server's operating system for the *data* string and "C" for the *type* string. For example, "'ABC' PURGE" "C" PKT on a local HP 48 would instruct an HP 48 server to purge variable *ABC*.

Related Commands: CLOSEIO, KERRM, SERVER

PMAX

PICT Maximum Command: Specifies (*x*, *y*) as the coordinates at the upper right corner of the display.

{}

Level 1	→	Level 1
(*x*, *y*)	→	

Keyboard Access: None. Must be typed in.

Affected by Flags: None

Remarks: The complex number (*x*, *y*) is stored as the second element in the reserved variable *PPAR*.

Related Commands: PDIM, PMIN, XRNG, YRNG

PMIN

PICT Minimum Command: Specifies (*x*, *y*) as the coordinates at the lower left corner of the display.

{}

Level 1	→	Level 1
(*x*,*y*)	→	

Keyboard Access: None. Must be typed in.

Affected by Flags: None

Remarks: The complex number (*x*, *y*) is stored as the first element in the reserved variable *PPAR*.

Related Commands: PDIM, PMAX, XRNG, YRNG

POLAR

Polar Plot Type Command: Sets the plot type to POLAR.

Keyboard Access: ◀ PLOT TYPE FCN AXES

Affected by Flags: Simultaneous Plotting (−28), Curve Filling (−31)

Remarks: When the plot type is POLAR, the DRAW command plots the current equation in polar coordinates, where the independent variable is the polar angle and the dependent variable is the radius. The current equation is specified in the reserved variable *EQ*.

The plotting parameters are specified in the reserved variable *PPAR*, which has this form:

$\{ (x_{min}, y_{min}) (x_{max}, y_{max})$ *indep res axes ptype depend* $\}$

For plot type POLAR, the elements of *PPAR* are used as follows:

- (x_{min}, y_{min}) is a complex number specifying the lower left corner of *PICT* (the lower left corner of the display range). The default value is (−6.5,−3.1).
- (x_{max}, y_{max}) is a complex number specifying the upper right corner of *PICT* (the upper right corner of the display range). The default value is (6.5,3.2).
- *indep* is a name specifying the independent variable, or a list containing such a name and two numbers specifying the minimum and maximum values for the independent variable (the plotting range). The default value of *indep* is *X*.
- *res* is a real number specifying the interval, in user-unit coordinates, between values of the independent variable. The default value is 0, which specifies an interval of 2 degrees, 2 grads, or $\pi/90$ radians.
- *axes* is a list containing one or more of the following, in the order listed: a complex number specifying the user-unit coordinates of the plot origin, a list specifying the tick-mark annotation, and two strings specifying labels for the horizontal and vertical axes. The default value is (0,0).
- *ptype* is a command name specifying the plot type. Executing the command POLAR places the name POLAR in *ptype*.

- *depend* is a name specifying a label for the vertical axis. The default value is *Y*.

The current equation is plotted as a function of the variable specified in *indep*. The minimum and maximum values of the independent variable (the plotting range) can be specifed in *indep*; otherwise, the default minimum value is 0 and the default maximum value corresponds to one full circle in the current angle mode (360 degrees, 400 grads, or 2π radians). Lines are drawn between plotted points unless flag −31 is set.

If flag −28 is set, all equations are plotted simultaneously.

If *EQ* contains an expression or program, the expression or program is evaluated in Numerical Results mode for each value of the independent variable to give the values of the dependent variable. If *EQ* contains an equation, the plotting action depends on the form of the equation.

Form of Current Equation	Plotting Action
'*expr*=*expr*'	Each expression is plotted separately. The intersection of the two graphs shows where the expressions are equal.
'*name*=*expr*'	Only the expression is plotted.

Related Commands: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

POS

Position Command: Returns the position of a substring within a string or the position of an object within a list.

Level 2	Level 1	→	Level 1
"string"	*"substring"*	→	*n*
{ list }	*obj*	→	*n*

Keyboard Access:

◀ CHARS POS

PRG LIST ELEM POS

Affected by Flags: None

Remarks: If there is no match for *obj* or *substring*, POS returns zero.

Related Commands: CHR, NUM, REPL, SIZE, SUB

PREDV

Predicted y-Value Command: Returns the predicted dependent-variable value $y_{dependent}$, based on the independent-variable value $x_{independent}$, the currently selected statistical model, and the current regression coefficients in the reserved variable ΣPAR.

{}

Level 1	→	Level 1
$x_{independent}$	→	$y_{dependent}$

Keyboard Access: None. Must be typed in.

Remarks: Provided for compatibility with the HP 28. PREDV is the same as PREDY. See PREDY.

PREDX

Predicted x-Value Command: Returns the predicted independent-variable value $x_{independent}$, based on the dependent-variable value $y_{dependent}$, the currently selected statistical model, and the current regression coefficients in the reserved variable ΣPAR.

{}

Level 1	→	Level 1
$y_{dependent}$	→	$x_{independent}$

Keyboard Access: ◀ STAT FIT PREDX

Affected by Flags: None

Remarks: The value is predicted using the regression coefficients most recently computed with LR and stored in the reserved variable ΣPAR. For the linear statistical model, the equation used is this:

$$y_{dependent} = (mx_{independent}) + b$$

where m is the slope (the third element in ΣPAR) and b is the intercept (the fourth element in ΣPAR).

For the other statistical models, the equations used by PREDX are listed in the LR entry.

If PREDX is executed without having previously generated regression coefficients in ΣPAR, a default value of zero is used for both regression coefficients, and an error results.

Example: Given five columns of data in ΣDAT, the command sequence:

```
2 XCOL 5 YCOL LOGFIT LR 23 PREDX
```

sets column 2 as the independent variable column, sets column 5 as the dependent variable column, and sets the logarithmic statistical

model. It then executes LR, generating intercept and slope regression coefficients, and storing them in ΣPAR. Then, given a dependent value of 23, it returns a predicted independent value based on the regression coefficients and the statistical model.

Related Commands: COLΣ, CORR, COV, EXPFIT, ΣLINE, LINFIT, LOGFIT, LR, PREDY, PWRFIT, XCOL, YCOL

PREDY

Predicted y-Value Command: Returns the predicted dependent-variable value $y_{dependent}$, based on the independent-variable value $x_{independent}$, the currently selected statistical model, and the current regression coefficients in the reserved variable ΣPAR.

{}

Level 1	→	Level 1
$x_{independent}$	→	$y_{dependent}$

Keyboard Access: ◀ STAT F1 PREDY

Affected by Flags: None

Remarks: The value is predicted using the regression coefficients most recently computed with LR and stored in the reserved variable ΣPAR. For the linear statistical model, the equation used is this:

$$y_{dependent} = (mx_{independent}) + b$$

where m is the slope (the third element in ΣPAR) and b is the intercept (the fourth element in ΣPAR).

For the other statistical models, the equations used by PREDY are listed in the LR entry.

If PREDY is executed without having previously generated regression coefficients in ΣPAR, a default value of zero is used for both regression coefficients—in this case PREDY will return 0 for statistical models LINFIT and LOGFIT, and error for statistical models EXPFIT and PWRFIT.

Example: Given four columns of data in ΣDAT, the command sequence:

```
2 XCOL 4 YCOL PWRFIT LR 11 PREDY
```

sets column 2 as the independent variable column, sets column 4 as the dependent variable column, and sets the power statistical model. It then executes LR, generating intercept and slope regression coefficients, and storing them in ΣPAR. Then, given an independent value of 11, it returns a predicted dependent value based on the regression coefficients and the statistical model.

Related Commands: COLΣ, CORR, COV, EXPFIT, ΣLINE, LINFIT, LOGFIT, LR, PREDX, PWRFIT, XCOL, YCOL

PRLCD

Print LCD Command: Prints a pixel-by-pixel image of the current display (excluding the annunciators).

Keyboard Access: ◀ I/O PRINT PRLCD

Affected by Flags: Printing Device (−34), I/O Device (−33), Linefeed (−38)

If flag −34 is set (printer output directed to the serial port), flag −33 must be clear.

Flag −38 must be clear.

Remarks: The width of the printed image of characters in the display is narrower using PRLCD than using a print command such as PRI. The difference results from the spacing between characters. On the display there is a single blank column between characters, and PRLCD prints this spacing. Print commands such as PRI print two blank columns between adjacent characters.

Example: The command sequence ERASE DRAW PRLCD clears $PICT$, plots the current equation, then prints the graphics display.

Related Commands: CR, DELAY, OLDPRT, PRST, PRSTC, PRVAR, PRI

PROMPT

Prompt Command: Displays the contents of "*prompt*" in the status area, and halts program execution.

{}

Level 1	→	Level 1
"*prompt*"	→	

Keyboard Access: PRG NXT I/O NXT PROMPT

Affected by Flags: None

Remarks: PROMPT is equivalent to 1 DISP 1 FREEZE HALT.

Related Commands: CONT, DISP, FREEZE, HALT, INFORM, INPUT, MSGBOX

PROOT

Polynomial Roots Command: Returns all roots of an n-degree polynomial having real or complex coefficients.

{}

Level 1	→	Level 1
[*array*]$_{\text{coefficients}}$	→	[*array*]$_{\text{roots}}$

Keyboard Access: ← SOLVE POLY PROOT

Affected by Flags: Infinite Result Exception (−22)

Remarks: For an n^{th}-order polynomial, the argument must be a real or complex array of length $n+1$ containing the coefficients listed from highest order to lowest. The result is a real or complex vector of length n containing the computed roots.

PROOT interprets leading coefficients of zero in a limiting sense. As a leading coefficient approaches zero, a root of the polynomial approaches infinity: therefore, if flag −22 is clear (the default), PROOT reports an Infinite Result error if a leading coefficient is zero. If flag −22 is set, PROOT returns a root of (MAXREAL,0) for each leading zero in an array containing real coefficients, and a root of (MAXREAL,MAXREAL) for each leading zero in an array containing complex coefficients.

Example: Find the roots of the polynomial $x^4+2x^3-25x^2-26x+120$:

`[ 1 2 -25 -26 120 ] PROOT` returns `[ 2 -3 4 -5 ]`.

Related Commands: PCOEF, PEVAL

PRST

Print Stack Command: Prints all objects in the stack, starting with the object in the highest level.

Keyboard Access: ← I/O PRINT PRST

Affected by Flags: Double-Spaced Printing (−37), Printing Device (−34), I/O Device (−33), Linefeed (−38)

If flag −34 is set (printer output directed to the serial port), flag −33 must be clear.

When flag −38 is set, linefeeds are *not* added at the end of each print line. Generally, flag −38 should be clear for execution of PRST. PRST leaves the stack unchanged.

Remarks: Objects are printed in multiline printer format. See the PR1 entry for a description of multiline printer format.

Related Commands: CR, DELAY, OLDPRT, PRLCD, PRSTC, PRVAR, PR1

PRSTC

Print Stack (Compact) Command: Prints in compact form all objects in the stack, starting with the object in the highest level.

Keyboard Access: ← I/O PRINT PRSTC

Affected by Flags: Double-Spaced Printing (−37), Printing Device (−34), I/O Device (−33), Linefeed (−38)

If flag −34 is set (printer output directed to the serial port), flag −33 must be clear.

When flag −38 is set, linefeeds are *not* added at the end of each print line. Generally, flag −38 should be clear for execution of PRSTC.

Remarks: Compact printer format is the same as compact display format. Multiline objects are truncated and appear on one line only. PRSTC leaves the stack unchanged.

Related Commands: CR, DELAY, OLDPRT, PRLCD, PRST, PRVAR, PR1

PRVAR

Print Variable Command: Searches the current directory path or port for the specified variables and prints the name and contents of each variable.

Level 1	→	Level 1
'name'	→	
{ name$_1$ name$_2$ ... }	→	
:n$_{port}$:'global'	→	

Keyboard Access: ← I/O PRINT PRVAR

Affected by Flags: Double-Spaced Printing (−37), Printing Device (−34), I/O Device (−33), Linefeed (−38)

If flag −34 is set (printer output directed to the serial port), flag −33 must be clear.

When flag −38 is set, linefeeds are *not* added at the end of each print line. Generally, flag −38 should be clear for execution of PRVAR.

Remarks: Objects are printed in multiline printer format. See the PR1 entry for a description of multiline printer format.

Related Commands: CR, DELAY, OLDPRT, PR1, PRLCD, PRST, PRSTC

PR1

Print Level 1 Command: Prints an object in multiline printer format.

Keyboard Access: ← I/O PR1

Affected by Flags: Double-Spaced Printing (−37), Printing Device (−34), I/O Device (−33), Linefeed (−38)

If flag −34 is set (printer output directed to the serial port), flag −33 must be clear.

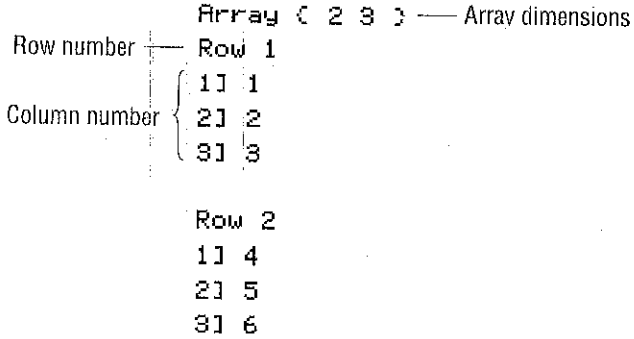
Remarks: All objects except strings are printed with their identifying delimiters. Strings are printed without the leading and trailing " delimiters. PR1 leaves the stack unchanged.

Multiline printer format is similar to multiline display format, with the following exceptions:

- Strings and names that are more than 24 characters long are continued on the next printer line.
- The real and imaginary parts of complex numbers are printed on separate lines if they don't fit on the same line.
- Arrays are printed with a numbered heading for each row and with a column number before each element. For example, the 2 × 3 array

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

would be printed as follows:



Related Commands: CR, DELAY, OLDPRT, PRLCD, PRST, PRSTC, PRVAR

PSDEV

Population Standard Deviation Command: Calculates the population standard deviation of each of the m columns of coordinate values in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	x_{psdev}
	→	[x_{psdev1} x_{psdev2} ··· x_{psdevm}]

Keyboard Access: ◀ STAT VAR NXT PSDEV

Affected by Flags: None

Remarks: PSDEV returns a vector of m real numbers, or a single real number if $m = 1$. The population standard deviation is computed using this formula:

$$\sqrt{\frac{1}{n}\sum_{k=1}^{n}(x_k - \overline{x})^2}$$

where x_k is the kth coordinate value in a column, $\overline{x}$ is the mean of the data in this column, and n is the number of data points.

Related Commands: MEAN, PCOV, PVAR, SDEV, TOT, VAR

PURGE

Purge Command: Purges the named variables or empty subdirectories from the current directory.

Level 1	→	Level 1
'global'	→	
{ $global_1$ ··· $global_n$ }	→	
PICT	→	
:n_{port} :$name_{backup}$	→	
:n_{port} :$n_{library}$	→	

Keyboard Access: ◀ PURGE

Affected by Flags: None

Remarks: PURGE executed in a program does not save its argument for recovery by LASTARG.

To empty a named directory before purging it, use PGDIR.

To help prepare a list of variables for purging, use VARS.

Purging *PICT* replaces the current graphics object with a 0 × 0 graphics object.

If a list of objects (with global names, backup objects, library objects, or *PICT*) for purging contains an invalid object, then the objects

preceding the invalid object are purged, and the error Bad Argument Type occurs.

To purge a library or backup object, tag the library number or backup name with the appropriate port number (: n_{port}), which must be in the range from 0 to 33. (A library can be purged from RAM only.) For a backup object, the port number can be replaced with the wildcard character &, in which case the HP 48 will search ports 33 through 0, and then main memory for the named backup object.

Library objects in RAM can be purged, while those in ROM (application cards and write-protected RAM cards) cannot. A library object must be detached before it can be purged from the *HOME* directory.

Neither a library object nor a backup object can be purged if it is currently "referenced" internally by stack pointers (such as an object on the stack, in a local variable, on the LAST stack, or on an internal return stack). This produces the error Object in Use. To avoid these restrictions, use NEWOB before purging. (See NEWOB.)

Related Commands: CLEAR, CLUSR, CLVAR, NEWOB, PGDIR

PUT

Put Element Command: In the level 3 array or list, PUT replaces with z_{put} or obj_{put} the object whose position is specified in level 2; if the array or list is unnamed, returns the new array or list.

Level 3	Level 2	Level 1	→	Level 1
[[*matrix*]]$_1$	$n_{position}$	z_{put}	→	[[*matrix*]]$_2$
[[*matrix*]]$_1$	{ n_{row} m_{col} }	z_{put}	→	[[*matrix*]]$_2$
'$name_{matrix}$'	$n_{position}$	z_{put}	→	
'$name_{matrix}$'	{ n_{row} m_{col} }	z_{put}	→	
[*vector*]$_1$	$n_{position}$	z_{put}	→	[*vector*]$_2$
[*vector*]$_1$	{ $n_{position}$ }	z_{put}	→	[*vector*]$_2$
'$name_{vector}$'	$n_{position}$	z_{put}	→	
'$name_{vector}$'	{ $n_{position}$ }	z_{put}	→	
{ *list* }$_1$	$n_{position}$	obj_{put}	→	{ *list* }$_2$
{ *list* }$_1$	{ $n_{position}$ }	obj_{put}	→	{ *list* }$_2$
'$name_{list}$'	$n_{position}$	obj_{put}	→	
'$name_{list}$'	{ $n_{position}$ }	obj_{put}	→	

Keyboard Access: (PRG) LIST ELEM PUT

Affected by Flags: None

Remarks: For matrices, $n_{position}$ counts in row order.

If the argument in level 3 is a name, PUT alters the named array or list and returns nothing to the stack.

Examples: This command sequence:
`[[ 2 3 4 ][ 4 i 2 ]] ( i 3 ) 96 PUT` returns
`[[ 2 3 96 ][ 4 i 2 ]]`.

The command sequence `[[ 2 3 4 ][ 4 i 2 ]] 5 96 PUT` returns
`[[ 2 3 4 ][ 4 96 2 ]]`.

The command sequence `( A B C D E ) ( 3 ) 'Z' PUT` returns
`( A B Z D E )`.

Related Commands: GET, GETI, PUTI

PUTI

Put and Increment Index Command: In the level 3 array or list, replaces with z_{put} or obj_{put} the object whose position is specified in level 2, returning the new array or list *and* the next position in that array or list.

Level 3	Level 2	Level 1	→	Level 2	Level 1
[[*matrix*]]$_1$	n_{pos1}	z_{put}	→	[[*matrix*]]$_2$	n_{pos2}
[[*matrix*]]$_1$	{ n_r m_c }$_1$	z_{put}	→	[[*matrix*]]$_2$	{ n_r m_c }$_2$
'$name_{matrix}$'	n_{pos1}	z_{put}	→	'$name_{matrix}$'	n_{pos2}
'$name_{matrix}$'	{ n_r m_c }$_1$	z_{put}	→	'$name_{matrix}$'	{ n_r m_c }$_2$
[*vector*]$_1$	n_{pos1}	z_{put}	→	[*vector*]$_2$	n_{pos2}
[*vector*]$_1$	{ n_{pos1} }	z_{put}	→	[*vector*]$_2$	{ n_{pos2} }
'$name_{vector}$'	n_{pos}	z_{put}	→	'$name_{vector}$'	n_{pos2}
'$name_{vector}$'	{ n_{pos1} }	z_{put}	→	'$name_{vector}$'	{ n_{pos2} }
{ *list* }$_1$	n_{pos1}	obj_{put}	→	{ *list* }$_2$	n_{pos2}
{ *list* }$_1$	{ n_{pos1} }	obj_{put}	→	{ *list* }$_2$	{ n_{pos2} }
'$name_{list}$'	n_{pos1}	obj_{put}	→	'$name_{list}$'	n_{pos2}
'$name_{list}$'	{ n_{pos1} }	obj_{put}	→	'$name_{list}$'	{ n_{pos2} }

Keyboard Access: PRG LIST ELEM PUTI

Affected by Flags: Index Wrap Indicator (−64)

The Index Wrap Indicator flag is cleared on each execution of PUTI *until* the position (index) wraps to the first position in the array or list, at which point the flag is set. The next execution of PUTI again clears the flag.

Remarks: For matrices, the position is incremented in *row* order.

Unlike PUT, PUTI returns a named array or list (to level 2). This enables a subsequent execution of PUTI at the next position of a named array or list.

Example: The following program uses PUTI and flag −64 to replace A, B, and C in the list with X.


```
« { A B C } DO 'X' PUTI UNTIL -64 FS? END »
```

Related Commands: GET, GETI, PUT

PVAR

Population Variance Command: Calculates the population variance of the coordinate values in each of the m columns in the current statistics matrix (ΣDAT).

Level 1	→	Level 1
	→	$x_{pvariance}$
	→	[$x_{pvariance1}$ ⋯ $x_{pvariancem}$]

Keyboard Access: ◀ STAT 1-VAR NXT PVAR

Affected by Flags: None

Remarks: The population variance (equal to the square of the population standard deviation) is returned as a vector of m real numbers, or as a single real number if $m = 1$. The population variances are computed using this formula:

$$\frac{1}{n}\sum_{k=1}^{n}(x_k - \overline{x})^2$$

where x_k is the kth coordinate value in a column, $\overline{x}$ is the mean of the data in this column, and n is the number of data points.

Related Commands: MEAN, PCOV, PSDEV, SDEV, VAR

PVARS

Port-Variables Command: Returns a list of the backup objects ($:n_{port}:name$) and the library objects ($:n_{port}:n_{library}$) in the specified port. Also returns the available memory size (if RAM) or the memory type.

{ }

Level 1	→	Level 2	Level 1
n_{port}	→	{ $:n_{port}:name_{backup}$ ⋯ }	*memory*
n_{port}	→	{ $:n_{port}:n_{library}$ ⋯ }	*memory*

Keyboard Access: ◀ LIBRARY PVARS

Affected by Flags: None

Remarks: The port number, n_{port}, must be in the range from 0 to 33.

- If $n_{port} = 0$, then *memory* is bytes of available main RAM.
- If the port contains independent RAM, then *memory* is bytes of available RAM in that port.
- If the port contains merged RAM, then *memory* is "SYSRAM".
- If the port contains ROM, then *memory* is "ROM"
- If the port is empty, then the message Port Not Available appears.

Related Commands: PVARS, VARS

PVIEW

PICT View Command: Displays *PICT* with the specified coordinate at the upper left corner of the graphics display.

Level 1	→	Level 1
(*x*, *y*)	→	
{ #*n* #*m* }	→	
{ }	→	

Keyboard Access:

PRG NXT I/O PVIEW

PRG PICT NXT PVIEW

Affected by Flags: None

Remarks: *PICT* must fill the entire display on execution of PVIEW. Thus, if a position other than the upper left corner of *PICT* is specified, *PICT* must be large enough to fill a rectangle that extends 131 pixels to the right and 64 pixels down.

If PVIEW is executed from a program with a coordinate argument (versus an empty list), the graphics display persists only until the keyboard is ready for input (for example, until the end of program execution). However, the FREEZE command freezes the display until a key is pressed.

If PVIEW is executed with an *empty* list argument, *PICT* is centered in the graphics display with scrolling mode activated. In this case, the graphics display persists until CANCEL is pressed.

PVIEW does *not* activate the graphics cursor or the Picture menu. To activate the graphics cursor and Picture menu, execute PICTURE.

Example: The program

```
« { # 0d # 0d } PVIEW  7 FREEZE »
```

displays *PICT* in the graphics display with coordinates { # 0d # 0d } in the upper left corner of the display, then freezes the full display until a key is pressed.

PVIEW

Related Commands: FREEZE, PICTURE, TEXT

PWRFIT

Power Curve Fit Command: Stores PWRFIT as the fifth parameter in the reserved variable ΣPAR, indicating that subsequent executions of LR are to use the power curve fitting model.

Keyboard Access: ◀ STAT 2-PAR MODL PWRFT

Affected by Flags: None

Remarks: LINFIT is the default specification in ΣPAR. For a description of ΣPAR, see appendix D, "Reserved Variables."

Related Commands: BESTFIT, EXPFIT, LINFIT, LOGFIT, LR

PX→C

Pixel to Complex Command: Converts the specified pixel coordinates to user-unit coordinates.

Level 1	→	Level 1
{ #*n* #*m* }	→	(*x*, *y*)

Keyboard Access: PRG PLOT NXT PX→C

Affected by Flags: None

Remarks: The user-unit coordinates are derived from the (x_{min}, y_{min}) and (x_{max}, y_{max}) parameters in the reserved variable *PPAR*. The coordinates correspond to the geometrical center of the pixel.

Related Commands: C→PX

→Q

To Quotient Command: Returns a rational form of the argument.

{ }

Level 1	→	Level 1
x	→	'*a*/*b*'
(*x*,*y*)	→	'*a*/*b*+*c*/*d**i'
'*symb*$_1$'	→	'*symb*$_2$'

Keyboard Access: ◀ SYMBOLIC NXT →Q

Affected by Flags: Number Display (−45 to −50)

Remarks: The rational result is a "best guess", since there might be more than one rational expression consistent with the argument. →Q finds a quotient of integers that agrees with the argument to within the number of decimal places specified by the display format mode.

→Q also acts on numbers that are part of algebraic expressions or equations.

Example: 'Y+2.5' →Q returns 'Y+5/2'.

Related Commands: →Qπ, /

→Qπ

To Quotient Times π Command: Returns a rational form of the argument, *or* a rational form of the argument with π factored out, whichever yields the smaller denominator.

{}

Level 1	→	Level 1
x	→	'*a/b**π'
x	→	'*a/b*'
'*symb*$_1$'	→	'*symb*$_2$'
(*x*,*y*)	→	'*a/b**π+*c/d**π*i'
(*x*,*y*)	→	'*a/b*+*c/d**i'

Keyboard Access: ◀ SYMBOLIC NXT →Qπ

Affected by Flags: Number Format (−45 to −50)

Remarks: →Qπ computes two quotients (rational expressions) and compares them: the quotient of the argument, and the quotient of the argument divided by π. It returns the fraction with the smaller denominator; if the argument was divided by π, then π is a factor in the result.

The rational result is a "best guess," since there might be more than one rational expression consistent with the argument. →Qπ finds a quotient of integers that agrees with the argument to the number of decimal places specified by the display format mode.

→Qπ also acts on numbers that are part of algebraic expressions or equations.

For a complex argument, the real or imaginary part (or both) can have π as a factor.

Example: In Fix mode to four decimal places, `6.2832 →Qπ` returns `'2*π'` In Standard mode, however, `6.2832 →Qπ` returns `3927/625`.

Related Commands: →Q, /, π

QR

QR Factorization of a Matrix Command: Returns the QR factorization of an $n \times m$ matrix.

{}

Level 1	→	Level 3	Level 2	Level 1
[[*matrix*]]$_A$	→	[[*matrix*]]$_Q$	[[*matrix*]]$_R$	[[*matrix*]]$_P$

Keyboard Access: MTH MATR FACTR QR

Affected by Flags: None

Remarks: QR factors $m \times n$ matrix A into three matrices:

- Q is an $m \times m$ orthogonal matrix.
- R is an $m \times n$ upper trapezoidal matrix.
- P is a $n \times n$ permutation matrix.

Where $A \times P = Q \times R$.

Related Commands: LQ, LSQ

QUAD

Solve Quadratic Equation Command: Solves an algebraic object '*symb*$_1$' for the variable *global*, and returns an expression '*symb*$_2$' representing the solution.

{}

Level 2	Level 1	→	Level 1
'*symb*$_1$'	'*global*'	→	'*symb*$_2$'

Keyboard Access: ◀ SYMBOLIC QUAD

Affected by Flags: Principal Solution (−1)

Remarks: QUAD calculates the second-degree Taylor series approximation of '*symb*$_1$' to convert it to quadratic form. The solution '*symb*$_2$' is exact if '*symb*$_1$' is second degree or less in *global*.

Since QUAD evaluates '*symb*$_1$', any variables in '*symb*$_1$' other than *global* should not exist in the current directory if they are to remain in the solution as formal variables.

QUAD generally does not work if *global* needs units to satisfy the equation.

Example: `'A*X^2+B*X+C=0' 'X' QUAD` returns
`'X=(-B+±1*√(B^2-4*(A*2/2)*C))/(2*(A*2/2))'`

which reduces to the familiar quadratic solution:

$$X = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Related Commands: COLCT, EXPAN, ISOL, SHOW

QUOTE

Quote Argument Function: Returns its argument unevaluated.

{}

Level 1	→	Level 1
'*symb*'	→	'*symb*'
obj	→	*obj*

Keyboard Access: ← SYMBOLIC NXT NXT QUOTE

Affected by Flags: None

Remarks: When an algebraic expression is evaluated, the arguments to a function in the expression are evaluated before the function. For example, when `'SIN(X)'` is evaluated, the name X is evaluated first, and the result is left on the stack as the argument for SIN.

This process creates a problem for functions that require symbolic arguments. For example, the function $\int$ requires as one of its arguments a name specifying the variable of integration. If evaluating an integral expression caused the name to be evaluated, the result of evaluation would be left on the stack for $\int$, rather than the name itself. To avoid this problem, the HP 48 automatically (and invisibly) quotes such arguments. When the quoted argument is evaluated, the unquoted argument is returned.

If a user-defined function takes symbolic arguments, those arguments must be quoted using QUOTE, as demonstrated in the following example.

Example: The following user-defined function *ArcLen* calculates the arc length of a function:

```
«
 → start end expr var
 «
  start end
  expr var ∂ SQ 1 + √
  var ∫
 »
»
```

ENTER ' ArcLen STO

To use this user-defined function in an algebraic expression, the symbolic arguments must be quoted:

```
'ArcLen(0,π,QUOTE(SIN(X)),QUOTE(X))'
```

Related Commands: APPLY, | (Where)

RAD

Radians Mode Command: Sets Radians angle mode.

Keyboard Access:

◀ RAD

◀ MODES ANGL RAD

Affected by Flags: None

Remarks: RAD sets flag −17 and clears flag −18, and displays the RAD annunciator.

In Radians angle mode, real-number arguments that represent angles are interpreted as radians, and real-number results that represent angles are expressed in radians.

Related Commands: DEG, GRAD

RAND

Random Number Command: Returns a pseudo-random number generated using a seed value, and updates the seed value.

Level 1	→	Level 1
	→	x_{random}

Keyboard Access: MTH NXT PROB RAND

Affected by Flags: None

Remarks: The HP 48 uses a linear congruential method and a seed value to generate a random number x_{random} in the range $0 \leq x < 1$. Each succeeding execution of RAND returns a value computed from a seed value based upon the previous RAND value. (Use RDZ to change the seed.)

Related Commands: COMB, PERM, RDZ, !

RANK

Matrix Rank Command: Returns the rank of a rectangular matrix.

{}

Level 1	→	Level 1
[[*matrix*]]	→	n_{rank}

Keyboard Access: MTH MATR NORM NXT RANK

Affected by Flags: Singular Value (−54)

Remarks: Rank is computed by calculating the singular values of the matrix and counting the number of nonnegligible values. If all computed singular values are zero, RANK returns zero. Otherwise RANK consults flag −54 as follows:

- If flag −54 is clear (the default), RANK counts all computed singular values that are less than or equal to 1.E−14 times the largest computed singular value.
- If flag −54 is set, RANK counts all nonzero computed singular values.

Related Commands: LQ, LSQ, QR

RANM

Random Matrix Command: Returns a matrix of specified dimensions that contains random integers in the range −9 through 9.

Level 1	→	Level 1
{ *m n* }	→	[[*random matrix*]]$_{m \times n}$
[[*matrix*]]$_{m \times n}$	→	[[*random matrix*]]$_{m \times n}$

Keyboard Access: MTH MATR MAKE RANM

RANM

Affected by Flags: None

Remarks: The probability of a particular nonzero digit occurring is 0.05; the probability of 0 occurring is 0.1.

Related Commands: RAND, RDZ

RATIO

Prefix Divide Function: Prefix form of / (divide) generated by the EquationWriter application.

{}

Level 2	Level 1	→	Level 1
z_1	z_2	→	$z_1 \ / \ z_2$
[*array*]	[[*matrix*]]	→	[[*array* × $matrix^{-1}$]]
[*array*]	*z*	→	[*array*/*z*]
z	'*symb*'	→	'*z*/*symb*'
'*symb*'	*z*	→	'*symb*/*z*'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ / $symb_2$'
#n_1	n_2	→	#n_3
n_1	#n_2	→	#n_3
#n_1	#n_2	→	#n_3
x_unit_1	y_unit_2	→	$(x/y)\_unit_1\ /unit_2$
x	*y_unit*	→	(*x*/*y*)_1/*unit*
x_unit	*y*	→	(*x*/*y*)_*unit*
'*symb*'	*x_unit*	→	'*symb*/*x_unit*'
x_unit	'*symb*'	→	'*x_unit*/*symb*'

Keyboard Access: None. Must be typed in.

Affected by Flags: None

Remarks: RATIO is identical to / (divide), except that, in algebraic syntax, RATIO is a *prefix* function, while / is an *infix* function. For example, 'RATIO(A,2)' is equivalent to 'A/2'

RATIO is generated internally by the EquationWriter application when ▲ is used to start a numerator. It provides no additional functionality to / and appears externally only in the string that the EquationWriter application leaves on the stack when → " " is pressed or when the calculator runs out of memory.

Related Commands: /

RCEQ

Recall from EQ Command: Returns the unevaluated contents of the reserved variable *EQ* from the current directory.

Level 1	→	Level 1
	→	obj_{EQ}

Keyboard Access:

- ← PLOT EQ
- ← PLOT NXT SIZE EQ
- ← PLOT → EQ (program-entry mode)
- ← PLOT NXT SIZE → EQ (program-entry mode)

Affected by Flags: None

Remarks: To recall the contents of *EQ* from a parent directory (when *EQ* doesn't exist in the current directory) evaluate the name *EQ*.

Related Commands: STEQ

RCI

Multiply Row by Constant Command: Multiplies row n of a matrix (or element n of a vector) by a constant x_{factor}, and returns the modified matrix.

{}

Level 3	Level 2	Level 1	→	Level 1
[[*matrix*]]$_1$	x_{factor}	$n_{rownumber}$	→	[[*matrix*]]$_2$
[*vector*]$_1$	x_{factor}	$n_{elementnumber}$	→	[*vector*]$_2$

Keyboard Access: (MTH) MATR ROW RCI

Affected by Flags: None

Remarks: RCI rounds the row number to the nearest integer, and treats vector arguments as column vectors.

Related Commands: RCIJ

RCIJ

Add Multiplied Row Command: Multiplies row i of a matrix by a constant x_{factor}, adds this product to row j of the matrix, and returns the modified matrix. Or, multiplies element i of a vector by a constant x_{factor}, adds this product to element j of the vector, and returns the modified vector.

{}

Level 4	Level 3	Level 2	Level 1	→	Level 1
[[*matrix*]]$_1$	x_{factor}	n_{rowi}	n_{rowj}	→	[[*matrix*]]$_2$
[*vector*]$_1$	x_{factor}	$n_{elementi}$	$n_{elementj}$	→	[*vector*]$_2$

Keyboard Access: (MTH) MATR ROW RCIJ

Affected by Flags: None

Remarks: RCIJ rounds the row numbers to the nearest integer, and treats vector arguments as column vectors.

Related Commands: RCI

RCL

Recall Command: Returns the unevaluated contents of a specified variable or plug-in object.

Level 1	→	Level 1
'*name*'	→	*obj*
PICT	→	*grob*
:n_{port} :$n_{library}$	→	*obj*
:n_{port} :$name_{backup}$	→	*obj*

Keyboard Access: (→) (RCL)

Affected by Flags: None

Remarks: RCL searches the entire current path, starting with the current directory. To search a different path, specify { *path name* }, where *path* is the new path to the variable *name*. The *path* subdirectory does not become the current subdirectory (unlike EVAL).

To recall a library or backup object, tag the library number or backup name with the appropriate port number (n_{port}), which must be an integer in the range 0 to 33. (A library can be recalled from RAM only.) Recalling a backup object brings a copy of its *contents* to the stack, not the entire backup object.

To search for a backup object, replace the port number with the wildcard character &, in which case the HP 48 will search (in order) ports 33 through 0, and the main memory for the named backup object.

Related Commands: STO

RCLALARM

Recall Alarm Command: Recalls a specified alarm.

{}

Level 1	→	Level 1
n_{index}	→	{ *date time* obj_{action} x_{repeat} }

Keyboard Access: ◀ TIME ALRM RCLAL

Affected by Flags: None

Remarks: obj_{action} is the alarm execution action. If an execution action was not specified, obj_{action} defaults to an empty string.

x_{repeat} is the repeat interval in clock ticks, where 1 clock tick equals 1/8192 second. If a repeat interval was not specified, the default is 0.

Related Commands: DELALARM, FINDALARM, STOALARM

RCLF

Recall Flags Command: Returns a list containing two 64-bit binary integers representing the states of the 64 system and user flags, respectively.

Level 1	→	Level 1
	→	{ $\#n_{system}$ $\#n_{user}$ }

Keyboard Access: ◀ MODES FLAG NXT RCLF

Affected by Flags: Binary Integer Wordsize (−5 through −10)

The current wordsize must be 64 bits (the default wordsize) to recall the states of all 64 user flags and 64 system flags. If the current wordsize is 32, for example, RCLF returns two 32-bit binary integers.

Remarks: A bit with value 1 indicates that the corresponding flag is set; a bit with value 0 indicates that the corresponding flag is clear. The rightmost (least significant) bit of $\#n_{system}$ and $\#n_{user}$ indicate the states of system flag −1 and user flag +1, respectively.

Used with STOF, RCLF lets a program that alters the state of a flag or flags during program execution preserve the pre-program-execution flag status.

Related Commands: STOF

RCLKEYS

Recall Key Assignments Command: Returns the current user key assignments. This includes an S if the standard definitions are active (not suppressed) for those keys without user key assignments.

Level 1	→	Level 1
	→	{ obj_1 x_{key1} ... obj_n x_{keyn} }
	→	{ S obj_1 x_{key1} ... obj_n x_{keyn} }

Keyboard Access: ◀ MODES KEYS RCLK

Affected by Flags: User-Mode Lock (−61) and User Mode (−62) affect the status of the user keyboard.

Remarks: The argument x_{key} is a real number of the form $rc.p$ specifying the key by its row number r, its column number c, and its plane (shift) p. (For a definition of plane, see the entry for ASN.)

Related Commands: ASN, DELKEYS, STOKEYS

RCLMENU

Recall Menu Number Command: Returns the menu number of the currently displayed menu.

Level 1	→	Level 1
	→	x_{menu}

Keyboard Access: ◀ MODES MENU RCLM

Affected by Flags: None

Remarks: x_{menu} has the form $mm.pp$, where mm is the menu number and pp is the page of the menu. See the MENU entry for a list of the HP 48 built-in menus and the corresponding menu numbers.

Executing RCLMENU when the current menu is a user-defined menu (built by TMENU) returns `0.01` (in 2 Fix mode), indicating "Last menu"

Example: If the third page of the PRG DSPL menu is currently active, RCLMENU returns `13.03` (in 2 Fix mode).

Related Commands: MENU, TMENU

RCLΣ

Recall Sigma Command: Returns the current statistics matrix (the contents of reserved variable ΣDAT) from the current directory.

Level 1	→	Level 1
	→	*obj*

Keyboard Access:

◀ PLOT NXT STAT ΣDAT

◀ STAT MATR ΣDAT

◀ PLOT NXT STAT ▶ ΣDAT (program-entry mode)

◀ STAT MATR ▶ ΣDAT (program-entry mode)

Affected by Flags: None

Remarks: To recall ΣDAT from a parent directory (when ΣDAT doesn't exist in the current directory), evaluate the name ΣDAT.

Related Commands: CLΣ, STOΣ, Σ+, Σ−

RCWS

Recall Wordsize Command: Returns the current wordsize in bits (1 through 64).

Level 1	→	Level 1
	→	*n*

Keyboard Access: MTH BASE NXT RCWS

Affected by Flags: Binary Integer Wordsize (−5 through −10), Binary Integer Base (−11, −12)

Related Commands: BIN, DEC, HEX, OCT, STWS

RDM

Redimension Array Command: Rearranges the elements of the argument according to specified dimensions.

Level 2	Level 1	→	Level 1
[*vector*]$_1$	{ $n_{elements}$ }	→	[*vector*]$_2$
[*vector*]	{ n_{rows} m_{cols} }	→	[[*matrix*]]
[[*matrix*]]	{ $n_{elements}$ }	→	[*vector*]
[[*matrix*]]$_1$	{ n_{rows} m_{cols} }	→	[[*matrix*]]$_2$
'*global*'	{ $n_{elements}$ }	→	
'*global*'	{ n_{rows} m_{cols} }	→	

Keyboard Access: MTH MATR MAKE RDM

Affected by Flags: None

Remarks: If the list contains a single number $n_{elements}$, the result is an n-element vector. If the list contains two numbers n_{rows} and m_{cols}, the result is an $n \times m$ matrix.

Elements taken from the argument vector or matrix preserve the same row order in the resulting vector or matrix. If the result is dimensioned to contain fewer elements than the argument vector or matrix, excess elements from the argument vector or matrix at the end of the row order are discarded. If the result is dimensioned to contain more elements than the argument vector or matrix, the additional elements in the result at the end of the row order are filled with zeros (0,0 if the argument is complex).

If the argument vector or matrix is specified by *global*, the result replaces the argument as the contents of the variable.

Examples: [2 4 6 8] { 2 2 } RDM returns [[2 4][6 8]].

[[2 3 4][1 6 9]] 8 RDM returns [2 3 4 1 6 9 0 0].

Related Commands: TRN

RDZ

Randomize Command: Uses a real number x_{seed} as a seed for the RAND command.

{}

Level 1	→	Level 1
x_{seed}	→	

Keyboard Access: MTH NXT PROB RDZ

Affected by Flags: None

Remarks: If the argument is 0, a random value based on the system clock is used as the seed.

Related Commands: COMB, PERM, RAND, !

RE

Real Part Function: Returns the real part of the argument.

{}

Level 1	→	Level 1
x	→	*x*
x_unit	→	*x*
(*x*, *y*)	→	*x*
[*R-array*]	→	[*R-array*]
[*C-array*]	→	[*R-array*]
'*symb*'	→	'RE(*symb*)'

Keyboard Access: MTH NXT CMPL RE

Affected by Flags: Numerical Results (−3)

Remarks: If the argument is a vector or matrix, RE returns a real array, the elements of which are equal to the real parts of the corresponding elements of the argument array.

Related Commands: C→R, IM, R→C

RECN

Receive Renamed Object Command: Prepares the HP 48 to receive a file from another Kermit device, and to store the file in a specified variable.

{}

Level 1	→	Level 1
'name'	→	
"name"	→	

Keyboard Access: ◀ I/O NXT RECN

Affected by Flags: I/O Device (−33), I/O Data Format (−35), RECV Overwrite (−36), I/O Messages (−39)

When an HP 48 sends an object, it automatically appends a header that tells a receiving HP 48 whether to use ASCII or binary mode. Flag −35 has an effect only if this header is not present.

Remarks: RECN is identical to RECV except that the name under which the received data is stored is specified in the stack.

Related Commands: BAUD, CKSM, CLOSEIO, FINISH, KERRM, KGET, PARITY, RECV, SEND, SERVER, TRANSIO

RECT

Rectangular Mode Command: Sets Rectangular coordinate mode.

Keyboard Access:

MTH VECTR NXT RECT

◀ MODES MATRX RECT

Affected by Flags: None

Remarks: RECT clears flags −15 and −16, and clears the R∠Z and R∠∠ annunciators.

In Rectangular mode, vectors are displayed as rectangular components. Therefore, a 3D vector would appear as $[X\ Y\ Z]$.

Related Commands: CYLIN, SPHERE

RECV

Receive Object Command: Instructs the HP 48 to look for a named file from another Kermit device. The received file is stored in a variable named by the sender.

Keyboard Access: ◀ I/O RECV

Affected by Flags: I/O Device (−33), I/O Data Format (−35), RECV Overwrite (−36), I/O Messages (−39)

When an HP 48 sends an object, it automatically appends a header that tells a receiving HP 48 whether to use ASCII or binary mode. Flag −35 has an effect only if this header is not present.

Remarks: Since the HP 48does not normally look for incoming Kermit files, you must use RECV to tell it to do so.

Related Commands: BAUD, CKSM, FINISH, KGET, PARITY, RECN, SEND, SERVER, TRANSIO

REPEAT

REPEAT Command: Starts loop clause in WHILE ... REPEAT ... END indefinite loop structure.

See the WHILE entry for syntax information.

Keyboard Access: PRG BRCH WHILE REPEA

Remarks: See the WHILE entry for more information.

Related Commands: END, WHILE

REPL

Replace Command: Replaces a portion of the level 3 target object with the level 1 object, beginning at a position specified in level 2.

Level 3	Level 2	Level 1	→	Level 1
[[*matrix*]]$_1$	$n_{position}$	[[*matrix*]]$_2$	→	[[*matrix*]]$_3$
[[*matrix*]]$_1$	{ n_{row} n_{column} }	[[*matrix*]]$_2$	→	[[*matrix*]]$_3$
[*vector*]$_1$	$n_{position}$	[*vector*]$_2$	→	[*vector*]$_3$
{ *list*$_{target}$ }	$n_{position}$	{ *list*$_1$ }	→	{ *list*$_{result}$ }
"*string*$_{target}$"	$n_{position}$	"*string*$_1$"	→	"*string*$_{result}$"
grob$_{target}$	{#*n* #*m*}	*grob*$_1$	→	*grob*$_{result}$
grob$_{target}$	(*x*,*y*)	*grob*$_1$	→	*grob*$_{result}$
PICT	{#*n* #*m*}	*grob*$_1$	→	
PICT	(*x*,*y*)	*grob*$_1$	→	

Keyboard Access:

- ↶ CHARS REPL
- PRG LIST REPL
- PRG GROB REPL
- MTH MATR MAKE NXT REPL

Affected by Flags: None

Remarks: For arrays, $n_{position}$ counts in row order. For matrices, $n_{position}$ specifies the new location of the upper left-hand element of the replacement matrix.

For graphics objects, the upper left corner of $grob_1$ is positioned at the user-unit or pixel coordinates (*x*,*y*) or (#*n* #*m*). From there, it overwrites a rectangular portion of $grob_{target}$ or *PICT*. If $grob_1$ extends past $grob_{target}$ or *PICT* in either direction, it is truncated in that direction. If the specified coordinate is not on the target graphics object, the target graphics object does not change.

Examples: [[i i i]] [i i i i] [i i i i]]
6 [[2 2]] [2 2]] REPL returns
[[i i i]] [i 2 2 i]] [i 2 2 i]].

(A B C D E) 2 (F G) REPL returns (A F G D E).

"ABCDE" 5 "FG" REPL returns "ABCDFG"

ERASE PICT (0,0) # 5d # 5d BLANK NEG REPL replaces a portion of *PICT* with a 5 × 5 graphics object, each of whose pixels is on (dark), and whose upper left corner is positioned at (0,0) in *PICT*.

Related Commands: CHR, GOR, GXOR, NUM, POS, SIZE, SUB

RES

Resolution Command: Specifies the resolution of mathematical and statistical plots, where the resolution is the interval between values of the independent variable used to generate the plot.

{}

Level 1	→	Level 1
$n_{Interval}$	→	
#$n_{Interval}$	→	

Keyboard Access: ↶ PLOT PPAR RES

Affected by Flags: None

Remarks: A real number $n_{interval}$ specifies the interval in user units. A binary integer $\#n_{interval}$ specifies the interval in pixels.

The resolution is stored as the fourth item in *PPAR*, with default value 0. The interpretation of the default value is summarized in the following table.

Plot Type	Default Interval
BAR	10 pixels (bar width = 10 pixel columns)
DIFFEQ	unlimited: step size is not constrained
FUNCTION	1 pixel (plots a point in every column of pixels)
CONIC	1 pixel (plots a point in every column of pixels)
TRUTH	1 pixel (plots a point in every column of pixels)
GRIDMAP	RES does not apply
HISTOGRAM	10 pixels (bin width = 10 pixel columns)
PARAMETRIC	[independent variable range in user units]/130
PARSURFACE	RES does not apply
PCONTOUR	RES does not apply
POLAR	2° 2 grads, or $\pi/90$ radians
SCATTER	RES does not apply
SLOPEFIELD	RES does not apply
WIREFRAME	RES does not apply
YSLICE	1 pixel (plots a point in every column of pixels)

Related Commands: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

RESTORE

Restore HOME Command: Replaces the current *HOME* directory with the specified backup copy ($:n_{port}:name_{backup}$) previously created by ARCHIVE.

{}

Level 1	→	Level 1
$:n_{port}:name_{backup}$	→	
backup	→	

Keyboard Access: ← MEMORY NXT RESTO

Affected by Flags: None

Remarks: The specified port number must be in the range 0 to 33. Ports 1 and 2 must be configured as independent RAM (see FREE).

To restore a *HOME* directory that was saved on a remote system using :IO:*name* ARCHIVE, put the backup object itself on the stack and execute RESTORE.

Example: To restore a *HOME* directory that was saved as the file *AUG1* on a remote system, execute `'AUG1' SEND` on the remote system, then execute the following on the HP 48:

```
RECV 'AUG1' RCL RESTORE
```

Related Commands: ARCHIVE

REVLIST

Reverse List Command: Reverses the order of the elements in a list.

Level 1	→	Level 1
{ obj_n ... obj_1 }	→	{ obj_1 ... obj_n }

Keyboard Access:

PRG LIST PROG REVLI

MTH LIST REVLI

Affected by Flags: None

Related Commands: ↑SORT

RKF

Solve for Initial Values (Runge-Kutta-Fehlberg) Command: Computes the solution to an initial value problem for a differential equation, using the Runge-Kutta-Fehlberg (4,5) method.

Level 3	Level 2	Level 1	→	Level 2	Level 1
{ *list* }	x_{tol}	x_{Tfinal}	→	{ *list* }	x_{tol}
{ *list* }	{ x_{tol} x_{hstep} }	x_{Tfinal}	→	{ *list* }	x_{tol}

Keyboard Access: ↶ SOLVE DIFFE RKF

Affected by Flags: None

Remarks: RKF solves y'(t)=f(t,y), where $y(t_0)=y_0$. The arguments and results are as follows:

- { *list* } contains three items in this order: the independent (t) and solution (y) variables, and the right-hand side of the differential equation (or a variable where the expression is stored).
- x_{tol} sets the absolute error tolerance. If a list is used, the first value is the absolute error tolerance and the second value is the initial candidate step size.
- x_{Tfinal} specifies the final value of the independent variable.

RKF repeatedly calls RKFSTEP as it steps from the initial value to x_{Tfinal}.

Example: Solve the following initial value problem for y(8), given that y(0) = 0:

$$y' = \frac{1}{(1+t^2)} - 2y^2 = f(t,y)$$

1. Store the independent variable's initial value, 0, in T.
2. Store the dependent variable's initial value, 0, in Y
3. Store the expression, $\frac{1}{(1+t^2)} - 2y^2$, in F.
4. Enter a list containing these three items: { T Y F }.
5. Enter the tolerance. Use estimated decimal place accuracy as a guideline for choosing a tolerance: 0.00001.
6. Enter the final value for the independent variable: 8.

   The stack should look like this:

   ```
   { T Y F }
   .00001
   8
   ```

7. Press RKF. (The calculation takes a moment.) The variable T now contains 8, and Y now contains the value .123077277659.

The actual answer is .123076923077, so the calculated answer has an error of approximately .00000035, well within the specified tolerance.

Related Commands: RKFERR, RKFSTEP, RRK, RRKSTEP, RSBERR

RKFERR

Error Estimate for Runge-Kutta-Fehlberg Method Command: Returns the absolute error estimate for a given step h when solving an initial value problem for a differential equation.

Level 2	Level 1	→	Level 4	Level 3	Level 2	Level 1
{ *list* }	*h*	→	{ *list* }	*h*	y_{delta}	*error*

Keyboard Access: ↩ SOLVE RKFERR

Affected by Flags: None

Remarks: The arguments and results are as follows:

- { *list* } contains three items in this order: the independent (t) and solution (y) variables, and the right-hand side of the differential equation (or a variable where the expression is stored).
- h is a real number that specifies the step.
- y_{delta} displays the change in solution for the specified step.
- *error* displays the absolute error for that step. A zero error indicates that the Runge-Kutta-Fehlberg method failed and that Euler's method was used instead.

The absolute error is the absolute value of the estimated error for a scalar problem, and the row (infinity) norm of the estimated error vector for a vector problem. (The latter is a bound on the maximum error of any component of the solution.)

Related Commands: RKF, RKFSTEP, RRK, RRKSTEP, RSBERR

RKFSTEP

Next Solution Step for RKF Command: Computes the next solution step (h_{next}) to an initial value problem for a differential equation.

Level 3	Level 2	Level 1	→	Level 3	Level 2	Level 1
{ *list* }	x_{tol}	*h*	→	{ *list* }	x_{tol}	h_{next}

Keyboard Access: ↩ SOLVE RKFSTEP

Affected by Flags: None

Remarks: The arguments and results are as follows:

- { *list* } contains three items in this order: the independent (t) and solution (y) variables, and the right-hand side of the differential equation (or a variable where the expression is stored).
- x_{tol} sets the tolerance value.
- h specifies the initial candidate step.
- h_{next} is the next candidate step.

The independent and solution variables must have values stored in them. RKFSTEP steps these variables to the next point upon completion.

Note that the actual step used by RKFSTEP will be less than the input value h if the global error tolerance is not satisfied by that value. If a stringent global error tolerance forces RKFSTEP to reduce its stepsize to the point that the Runge-Kutta-Fehlberg method fails, then RKFSTEP will use the Euler method to compute the next solution step and will consider the error tolerance satisfied. The Runge-Kutta-Fehlberg method will fail if the current independent variable is zero and the stepsize $\leq 1.3 \times 10^{-498}$ or if the variable is nonzero and the stepsize is 1.3×10^{-10} times its magnitude.

Related Commands: RKF, RKFERR, RRK, RRKSTEP, RSBERR

RL

Rotate Left Command: Rotates a binary integer one bit to the left.

{}

Level 1	→	Level 1
#n_1	→	#n_2

Keyboard Access: MTH BASE NXT BIT RL

Affected by Flags: Binary Integer Wordsize (−5 through −10), Binary Integer Base (−11, −12)

Remarks: The leftmost bit of #n_1 becomes the rightmost bit of #n_2.

Related Commands: RLB, RR, RRB

RLB

Rotate Left Byte Command: Rotates a binary integer one byte to the left.

{}

Level 1	→	Level 1
#n_1	→	#n_2

Keyboard Access: MTH BASE NXT BYTE RLB

Affected by Flags: Binary Integer Wordsize (−5 through −10), Binary Integer Base (−11, −12)

Remarks: The leftmost byte of #n_1 becomes the rightmost byte of #n_2. RLB is equivalent to executing RL eight times.

Related Commands: RL, RR, RRB

RND

Round Function: Rounds an object to a specified number of decimal places or significant digits, or to fit the current display format.

{}

Level 2	Level 1	→	Level 1
z_1	n_{round}	→	z_2
z	'$symb_{round}$'	→	'RND(z,$symb_{round}$)'
'$symb$'	n_{round}	→	'RND($symb$,n_{round})'
'$symb_1$'	'$symb_{round}$'	→	'RND($symb_1$,$symb_{round}$)'
[$array_1$]	n_{round}	→	[$array_2$]
x_unit	n_{round}	→	y_unit
x_unit	'$symb_{round}$'	→	'RND(x_unit,$symb_{round}$)'

Keyboard Access: MTH REAL NXT NXT RND

Affected by Flags: Numerical Results (−3)

Remarks: n_{round} (or $symb_{round}$ if flag −3 is set) controls how the level 2 argument is rounded, as follows:

n_{round} or $symb_{round}$	Effect on Level 2 Argument
0 through 11	Rounded to n decimal places.
−1 through −11	Rounded to n significant digits.
12	Rounded to the current display format.

For complex numbers and arrays, each real number element is rounded. For unit objects, the numerical part of the object is rounded.

Examples: `(4.5792,8.1275) 2 RND` returns `(4.58,8.13)`.

`[ 2.34907 3.96351 2.73453 ] -2 RND` returns `[ 2.3 4 2.7 ]`.

Related Commands: TRNC

RNRM

Row Norm Command: Returns the row norm (infinity norm) of its argument array.

Level 1	→	Level 1
[*array*]	→	x_{rownorm}

Keyboard Access: (MTH) MATR NORM RNRM

Affected by Flags: None

Remarks: The row norm is the maximum (over all rows) of the sums of the absolute values of all elements in each row. For a vector, the row norm is the largest absolute value of any of its elements.

Related Commands: CNRM, CROSS, DET, DOT

ROLL

Roll Objects Command: Moves the contents of a specified level to level 1, and rolls upwards the portion of the stack beneath the specified level.

Level n+1 ... Level 2	Level 1	→	Level n ... Level 2	Level 1
$obj_n \ldots obj_1$	n	→	$obj_{n-1} \ldots obj_1$	obj_n

Keyboard Access: (↰)(STACK) ROLL

Affected by Flags: None

Remarks: 3 ROLL is equivalent to ROT.

Related Commands: OVER, PICK, ROLLD, ROT, SWAP

ROLLD

Roll Down Command: Moves the contents of level 1 to a specified level, and rolls downwards the portion of the stack beneath the specified level.

Lvl n+1 ... Lvl 2	Lvl 1	→	Lvl n	Lvl n-1 ... Lvl 1
$obj_n \ldots obj_1$	n	→	obj_1	$obj_n \ldots obj_2$

Keyboard Access: (↰)(STACK) ROLL

Affected by Flags: None

Related Commands: OVER, PICK, ROLL, ROT, SWAP

ROOT

Root-Finder Command: Returns a real number x_{root} that is a value of the specified variable *global* for which the specified program or algebraic object most nearly evaluates to zero or a local extremum.

Level 3	Level 2	Level 1	→	Level 1
≪ *program* ≫	'*global*'	*guess*	→	x_{root}
≪ *program* ≫	'*global*'	{ *guesses* }	→	x_{root}
'*symb*'	'*global*'	*guess*	→	x_{root}
'*symb*'	'*global*'	{ *guesses* }	→	x_{root}

Keyboard Access: (↰)(SOLVE) ROOT ROOT

Affected by Flags: None

Remarks: ROOT is the programmable form of the HP Solve application.

guess is an initial estimate of the solution. ROOT produces an error if it cannot find a solution, returning the message Bad Guess(es) if one or more of the guesses lie outside the domain of the equation, or returns the message Constant? if the equation returns the same value at every sample point. ROOT does *not* return interpretive messages when a root is found.

ROT

Rotate Objects Command: Rotates the first three objects on the stack, moving the object in level 3 to level 1.

Level 3	Level 2	Level 1	→	Level 3	Level 2	Level 1
obj_3	obj_2	obj_1	→	obj_2	obj_1	obj_3

Keyboard Access: ◀ STACK ROT

Affected by Flags: None

Remarks: ROT is equivalent to 3 ROLL.

Related Commands: OVER, PICK, ROLL, ROLLD, SWAP

→ROW

Matrix to Rows Command: Transforms a matrix into a series of row vectors and returns the vectors and a row count, or transforms a vector into its elements and returns the elements and an element count.

{}

Level 1	→	Level n+1 ...	Level 2	Level 1
[[*matrix*]]	→	[*vector*]$_{row\ i}$	[*vector*]$_{row\ n}$	$n_{rowcount}$
[*vector*]	→	$element_1$	$element_n$	$n_{elementcount}$

Keyboard Access: MTH MATR ROW →ROW

Affected by Flags: None

Related Commands: →COL, COL→, ROW→

ROW+

Insert Row Command: Inserts an array into a matrix (or one or more numbers into a vector) at the position indicated by n_{index}, and returns the modified matrix (or vector).

{}

Level 3	Level 2	Level 1	→	Level 1
[[*matrix*]]$_1$	[*matrix*]$_2$	n_{index}	→	[[*matrix*]]$_3$
[[*matrix*]]$_1$	[*vector*]$_{row}$	n_{index}	→	[[*matrix*]]$_2$
[*vector*]$_1$	$n_{element}$	n_{index}	→	[*vector*]$_2$

Keyboard Access: MTH MATR ROW ROW+

Affected by Flags: None

Remarks: The inserted array must have the same number of columns as the target array.

n_{index} is rounded to the nearest integer. The original array is redimensioned to include the new columns or elements, and the elements at and below the insertion point are shifted down.

Related Commands: COL−, COL+, ROW−, RSWP

ROW−

Delete Row Command: Deletes row n of a matrix (or element n of a vector), and returns the modified matrix (or vector) and the deleted row (or element).

{}

Level 2	Level 1	→	Level 2	Level 1
[[*matrix*]]$_1$	n_{row}	→	[[*matrix*]]$_2$	[*vector*]$_{\text{row}}$
[*vector*]$_1$	n_{element}	→	[*vector*]$_2$	*element*$_n$

Keyboard Access: MTH MATR ROW ROW−

Affected by Flags: None

Remarks: n_{row} or n_{element} is rounded to the nearest integer.

Related Commands: COL−, COL+, ROW+, RSWP

ROW→

Rows to Matrix Command: Transforms a series of row vectors and a row count into a matrix containing those rows, or transforms a sequence of numbers and an element count into a vector with those numbers as elements.

Level$_{n+1}$ ...	Level 2	Level 1	→	Level 1
[*vector*]$_{\text{row 1}}$	[*vector*]$_{\text{row n}}$	n_{rowcount}	→	[[*matrix*]]
element$_1$	*element*$_n$	$n_{\text{elementcount}}$	→	[*vector*]$_{\text{column}}$

Keyboard Access: MTH MATR ROW ROW→

Affected by Flags: None

Related Commands: →COL, COL→, →ROW

RR

Rotate Right Command: Rotates a binary integer one bit to the right.

{}

Level 1	→	Level 1
#n_1	→	#n_2

Keyboard Access: MTH BASE NXT BIT RR

Affected by Flags: Binary Integer Wordsize (−5 through −10), Binary Integer Base (−11, −12)

Remarks: The rightmost bit of #n_1 becomes the leftmost bit of #n_2.

Related Commands: RL, RLB, RRB

RRB

Rotate Right Byte Command: Rotates a binary integer one byte to the right.

{}

Level 1	→	Level 1
#n_1	→	#n_2

Keyboard Access: MTH BASE NXT BYTE RRB

Affected by Flags: Binary Integer Wordsize (−5 through −10), Binary Integer Base (−11, −12).

Remarks: The rightmost byte of #n_1 becomes the leftmost byte of #n_2. RRB is equivalent to doing RR eight times.

Related Commands: RL, RLB, RR

RREF

Reduced Row Echelon Form Command: Converts a rectangular matrix to reduced row echelon form.

{}

Level 1	→	Level 1
[[*matrix*]]$_A$	→	[[*matrix*]]$_R$

Keyboard Access: (MTH) MATR MAKE RREF

Affected by Flags: Singular Values (−54)

Remarks: Converts a given matrix into reduced row echelon form. Since row echelon form is primarily used for studying systems of linear equations, RREF ignores very small pivots if system flag −54 is clear.

Related Commands: LU

RRK

Solve for Initial Values (Rosenbrock, Runge-Kutta) Command: Computes the solution to an intial value problem for a differential equation with known partial derivatives.

Level 3	Level 2	Level 1	→	Level 2	Level 1
{ *list* }	x_{tol}	x_{Tfinal}	→	{ *list* }	x_{tol}
{ *list* }	{ x_{tol} x_{hstep} }	x_{Tfinal}	→	{ *list* }	x_{tol}

Keyboard Access: (←) (SOLVE) DIFEQ RRK

Affected by Flags: None

Remarks: RRK solves y'(t)=f(t,y), where $y(t_0)=y_0$. The arguments and results are as follows:

- { *list* } contains five items in this order:
  - The independent variable (t).
  - The solution variable (y).
  - The right-hand side of the differential equation (or a variable where the expression is stored).
  - The partial derivative of y'(t) with respect to the solution variable) (or a variable where the expression is stored).
  - The partial derivative of y'(t) with respect to the independent variable (or a variable where the expression is stored).
- x_{tol} sets the tolerance value. If a list is used, the first value is the tolerance and the second value is the initial candidate step size.
- x_{Tfinal} specifies the final value of the independent variable.

RRK repeatedly calls RKFSTEP as it steps from from the initial value to x_{Tfinal}.

Example: Solve the following initial value problem for y(8), given that y(0) = 0:

$$y' = \frac{1}{(1+t^2)} - 2y^2 = f(t,y)$$

The derivative of the function with respect to y ($\partial f/\partial y$) is -4y, and the derivative of the function with respect to t ($\partial f/\partial t$) is $\frac{-2t}{(1+t^2)^2}$.

1. Store the independent variable's initial value, 0, in T.
2. Store the dependent variable's initial value, 0, in Y
3. Store the expression, $\frac{1}{(1+t^2)} - 2y^2$, in F
4. Store $\partial f/\partial y$, -4y, in FY.
5. Store $\partial f/\partial t$, $\frac{-2t}{(1+t^2)^2}$, in FT.
6. Enter these five items in a list: { T Y F FY FT }.
7. Enter the tolerance. Use estimated decimal place accuracy as a guideline for choosing a tolerance: 0.00001.
8. Enter the final value for the independent variable: 8.

The stack should look like this:

```
{ T Y F FY FT}
.00001
8
```

9. Press RRK. (The calculation takes a moment.) The variable T now contains 8, and Y now contains the value .123077277659.

The actual answer is .123076923077, so the calculated answer has an error of approximately .00000035, well within the specified tolerance.

Related Commands: RKF, RKFERR, RKFSTEP, RRKSTEP, RSBERR

RRKSTEP

Next Solution Step and Method (RKF or RRK) Command: Computes the next solution step (h_{next}) to an initial value problem for a differential equation, and displays the method used to arrive at that result.

Lvl 4	Lvl 3	Lvl 2	Lvl 1	→	Lvl 4	Lvl 3	Lvl 2	Lvl 1
{ *list* }	x_{tol}	h	*last*	→	{ *list* }	x_{tol}	h_{next}	*current*

Keyboard Access: ← SOLVE DIFFEQ RRKS

Affected by Flags: None

Remarks: The arguments and results are as follows:

- { *list* } contains five items in this order:
  - The independent variable (t).
  - The solution variable (y).
  - The right-hand side of the differential equation (or a variable where the expression is stored).
  - The partial derivative of y'(t) with respect to the solution variable) (or a variable where the expression is stored).

  - The partial derivative of y'(t) with respect to the independent variable (or a variable where the expression is stored).
- x_{tol} is the tolerance value.
- h specifies the initial candidate step.
- *last* specifies the last method used (RKF = 1, RRK = 2). If this is the first time you are using RRKSTEP, enter 0.
- *current* displays the current method used to arrive at the next step.
- h_{next} is the next candidate step.

The independent and solution variables must have values stored in them. RRKSTEP steps these variables to the next point upon completion.

Note that the actual step used by RRKSTEP will be less than the input value h if the global error tolerance is not satisfied by that value. If a stringent global error tolerance forces RRKSTEP to reduce its stepsize to the point that the Runge-Kutta-Fehlberg or Rosenbrock methods fails, then RRKSTEP will use the Euler method to compute the next solution step and will consider the error tolerance satisfied. The Rosenbrock method will fail if the current independent variable is zero and the stepsize $\leq 2.5 \times 10^{-499}$ or if the variable is nonzero and the stepsize is 2.5×10^{-11} times its magnitude. The Runge-Kutta-Fehlberg method will fail if the current independent variable is zero and the stepsize $\leq 1.3 \times 10^{-498}$ or if the variable is nonzero and the stepsize is 1.3×10^{-10} times its magnitude.

Related Commands: RKF, RKFERR, RKFSTEP, RRK, RSBERR

RSBERR

Error Estimate for Rosenbrock Method Command: Returns an error estimate for a given step h when solving an initial values problem for a differential equation.

Level 2	Level 1	→	Level 4	Level 3	Level 2	Level 1
{ *list* }	*h*	→	{ *list* }	*h*	y_{delta}	*error*

Keyboard Access: ◀ SOLVE DIFFE RSBERR

Affected by Flags: None

Remarks: The arguments and results are as follows:

- { *list* } contains five items in this order:
  - The independent variable (t).
  - The solution variable (y).
  - The right-hand side of the differential equation (or a variable where the expression is stored).
  - The partial derivative of y'(t) with respect to the solution variable) (or a variable where the expression is stored).
  - The partial derivative of y'(t) with respect to the independent variable (or a variable where the expression is stored).
- h is a real number that specifies the initial step.
- y_{delta} displays the change in solution.
- *error* displays the absolute error for that step. The *absolute* error is the absolute value of the estimated error for a scalar problem, and the row (infinity) norm of the estimated error vector for a vector problem. (The latter is a bound on the maximum error of any component of the solution.) A zero error indicates that the Rosenbrock method failed and Euler's method was used instead.

Related Commands: RKF, RKFERR, RKFSTEP, RRK, RRKSTEP

RSD

Residual Command: Computes the residual $\mathbf{B} - \mathbf{AZ}$ of the arrays **B**, **A**, and **Z**.

{ }

Level 3	Level 2	Level 1	→	Level 1
⌊ *vector* ⌋$_B$	[[*matrix*]]$_A$	[*vector*]$_Z$	→	[*vector*]$_{B-AZ}$
[[*matrix*]]$_B$	[[*matrix*]]$_A$	[[*matrix*]]$_Z$	→	[[*matrix*]]$_{B-AZ}$

Keyboard Access: MTH MATR NXT RSD

Affected by Flags: None

Remarks: **A**, **B**, and **Z** are restricted as follows:

- **A** must be a matrix.
- The number of columns of **A** must equal the number of elements of **Z** if **Z** is a vector, or the number of rows of **Z** if **Z** is a matrix.
- The number of rows of **A** must equal the number of elements of **B** if **B** is a vector, or the number of rows of **B** if **B** is a matrix.
- **B** and **Z** must both be vectors or both be matrices.
- **B** and **Z** must have the same number of columns if they are matrices.

RSD is typically used for computing a correction to **Z**, where **Z** has been obtained as an approximation to the solution **X** to the system of equations $\mathbf{AX} = \mathbf{B}$.

RSWP

Row Swap Command: Swaps rows i and j of a matrix and returns the modified matrix, or swaps elements i and j of a vector and returns the modified vector.

{}

Level 3	Level 2	Level 1	→	Level 1
$[[\ matrix\ ]]_1$	$n_{row\,i}$	$n_{row\,j}$	→	$[[\ matrix\ ]]_2$
$[\ vector\ ]_1$	$n_{element\,i}$	$n_{element\,j}$	→	$[\ vector\ ]_2$

Keyboard Access: (MTH) MATR ROW (NXT) RSWP

Affected by Flags: None

Remarks: Row numbers are rounded to the nearest integer. Vector arguments are treated as column vectors.

Related Commands: CSWP, ROW+, ROW−

R→B

Real to Binary Command: Converts a positive real integer to its binary integer equivalent.

{}

Level 1	→	Level 1
n	→	#n

Keyboard Access: (MTH) BASE R→B

Affected by Flags: Binary Integer Wordsize (−5 through −10), Binary Integer Base (−11, −12)

Remarks: For any value of $n \leq$ `#0`, the result is `#0`. For any value of $n \geq$ `1.8446744073709E19` (base 10), the result is `#FFFFFFFFFFFFFFFF` (base 16).

Related Commands: B→R

R→C

Real to Complex Command: Combines two real numbers or real arrays into a single complex number or complex array.

{}

Level 2	Level 1	→	Level 1
x	y	→	(x,y)
$[\ R\text{-}array_1\ ]$	$[\ R\text{-}array_2\ ]$	→	$[\ C\text{-}array\ ]$

Keyboard Access:

(PRG) TYPE (NXT) R→C

(MTH) (NXT) CMPL R→C

Affected by Flags: None

Remarks: The level 2 argument represents the real element(s) of the complex result. The level 1 argument represents the imaginary element(s) of the complex result.

Array arguments must have the same dimensions.

Related Commands: C→R, IM, RE

R→D

Radians to Degrees Function: Converts a real number expressed in radians to its equivalent in degrees.

{}

Level 1	→	Level 1
x	→	$(180/\pi)\ x$
'symb'	→	*'R→D(symb)'*

Keyboard Access: MTH REAL NXT NXT R→D

Affected by Flags: Numerical Results (−3)

Remarks: This function operates independently of the angle mode.

Related Commands: D→R

SAME

Same Object Command: Compares two objects, and returns a true result (1) if they are identical, and a false result (0) if they are not.

Level 2	Level 1	→	Level 1
obj_1	obj_2	→	0/1

Keyboard Access: PRG TEST NXT SAME

Affected by Flags: None

Remarks: SAME is identical in effect to == for all object types except algebraics, names, and some units. (For algebraics and names, == returns an expression that can be evaluated to produce a test result based on numerical values.)

Examples: « A B » (4,5) SAME returns 0.

« A B » « B A » SAME returns 0.

"CATS" "CATS" SAME returns 1.

Related Commands: TYPE, ==

SBRK

Serial Break Command: Interrupts serial transmission or reception.

Keyboard Access: ← I/O NXT SERIAL SBRK

Affected by Flags: I/O Device (−33)

Remarks: SBRK is typically used when a problem occurs in a serial data transmission.

Related Commands: BUFLEN, SRECV, STIME, XMIT

SCALE

Scale Plot Command: Adjusts the first two parameters in *PPAR*, (x_{min}, y_{min}) and (x_{max}, y_{max}), so that x_{scale} and y_{scale} are the new plot horizontal and vertical scales, and the center point doesn't change.

{}

Level 2	Level 1	→	Level 1
x_{scale}	y_{scale}	→	

Keyboard Access: ← PLOT PPAR NXT SCALE

Affected by Flags: None

Remarks: The scale in either direction is the number of user units per tick mark. The default scale in both directions is 1 user unit per tick mark.

Related Commands: AUTO, CENTR, *H, *W

SCATRPLOT

Draw Scatter Plot Command: Draws a scatterplot of (x, y) data points from the specified columns of the current statistics matrix (reserved variable ΣDAT).

Keyboard Access: ◀ STAT PLOT SCATR

Affected by Flags: None

Remarks: The data columns plotted are specified by XCOL and YCOL, and are stored as the first two parameters in the reserved variable ΣPAR. If no data columns are specified, columns 1 (independent) and 2 (dependent) are selected by default. The y-axis is autoscaled and the plot type is set to SCATTER.

When SCATRPLOT is executed from a program, the resulting display does not persist unless PICTURE or PVIEW is subsequently executed.

If PICTURE *is* subsequently executed, pressing STAT in the Picture environment draws a line to fit the data using the currently specified statistical model.

Example: The following program plots a scatter plot of the data in columns 3 and 4 of ΣDAT, draws a best fit line, and displays the plot:

```
« 3 XCOL 4 YCOL SCATRPLOT BESTFIT ΣLINE STEQ
  FUNCTION DRAW ( # 0d # 0d ) PVIEW 7 FREEZE »
```

Related Commands: BARPLOT, PICTURE, HISTPLOT, PVIEW, SCLΣ, XCOL, YCOL

SCATTER

Scatter Plot Type Command: Sets the plot type to SCATTER.

Keyboard Access: ◀ PLOT NXT STAT PTYPE SCATT

Affected by Flags: None

Remarks: When the plot type is SCATTER, the DRAW command plots points by obtaining x and y coordinates from two columns of the current statistics matrix (reserved variable ΣDAT). The columns are specified by the first and second parameters in the reserved variable ΣPAR (using the XCOL and YCOL commands). The plotting parameters are specified in the reserved variable $PPAR$, which has this form:

$$\{\ (x_{min}, y_{min})\ (x_{max}, y_{max})\ indep\ res\ axes\ ptype\ depend\ \}$$

For plot type SCATTER, the elements of $PPAR$ are used as follows:

- (x_{min}, y_{min}) is a complex number specifying the lower left corner of $PICT$ (the lower left corner of the display range). The default value is (-6.5,-3.).
- (x_{max}, y_{max}) is a complex number specifying the upper right corner of $PICT$ (the upper right corner of the display range). The default value is (6.5,3.2).
- *indep* is a name specifying the independent variable. The default value of *indep* is X.
- *res* is not used.
- *axes* is a list containing one or more of the following, in the order listed: a complex number specifying the user-unit coordinates of the plot origin, a list specifying the tick-mark annotation, and two strings specifying labels for the horizontal and vertical axes. The default value is (0,0).
- *ptype* is a command name specifying the plot type. Executing the command SCATTER places the name SCATTER in *ptype*.
- *depend* is a name specifying the dependent variable. The default value is Y

Related Commands: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE,

PCONTOUR, POLAR, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

SCHUR

Schur Decomposition of a Square Matrix Command: Returns the Schur decomposition of a square matrix.

{}

Level 1	→	Level 2	Level 1
[[*matrix*]]$_A$	→	[[*matrix*]]$_Q$	[[*matrix*]]$_T$

Keyboard Access: (MTH) MATR FACTR SCHUR

Affected by Flags: None

Remarks: SCHUR decomposes A into two matrices Q and T.

- If A is a complex matrix, Q is a unitary matrix, and T is an upper-triangular matrix.
- If A is a real matrix, Q is an orthogonal matrix, and T is an upper quasi-triangular matrix (T is upper block triangular with 1×1 or 2×2 diagonal blocks where the 2×2 blocks have complex conjugate eigenvalues).

In either case, $A \cong Q \times T \times \mathrm{TRN}(Q)$.

Related Commands: LQ, LU, QR, SVD, SVL, TRN

SCI

Scientific Mode Command: Sets the number display format to Scientific mode, which displays one digit to the left of the fraction mark and n significant digits to the right.

{}

Level 1	→	Level 1
n	→	

Keyboard Access: ← (MODES) FMT SCI

Affected by Flags: None

Remarks: Scientific mode is equivalent to scientific notation using $n+1$ significant digits, where $0 \le n \le 11$. (Values for n outside this range are rounded to the nearest integer.) In Scientific mode, numbers are displayed and printed like this:

(*sign*) *mantissa* E (*sign*) *exponent*

where the mantissa has the form $n.(n \ldots)$ and has zero to 11 decimal places, and the exponent has one to three digits.

Example: The number 103.6 in Scientific mode to four decimal places appears as 1.0360E2.

Related Commands: ENG, FIX, STD

SCLΣ

Scale Sigma Command: Adjusts (x_{min}, y_{min}) and (x_{max}, y_{max}) in *PPAR* so that a subsequent scatter plot exactly fills *PICT*.

Keyboard Access: None. Must be typed in.

Affected by Flags: None

Remarks: When the plot type is SCATTER, the command AUTO incorporates the functions of SCLΣ. In addition, the command