HP 49G Advanced Users Guide

Volume 1

Part D: Other Commands: Q to S

Introduction

This volume details the HP 49G commands and functions that are not computer algebra-specific. See part A, Computer algebra commands and functions, for information on computer algebra commands.

For each operation, the following details are provided:

Type: Function or command. Functions can be used as a part of an algebraic objects and commands cannot.

Description: A description of the operation.

Access: The menu or choose-list on which an operation can be found, and the keys that you press to access it. If the operation is on a sub-menu, the sub-menu name is in SMALL CAPITALS after the keys.

Input/Output: The input argument or arguments that the operation needs, and the outputs it produces.

See also: Related functions or commands

Q to R

→Q

Type: Command

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

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.

Access: CAT →Q

Input/Output:

Level 1/Argument 1		Level 1/Item 1
x	→	'a/b'
(x,y)	→	'$a/b + c/d*i$'
'$symb_1$'	→	'$symb_2$'

See also: →Qπ, /

→Qπ

Type: Command

Description: 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.

→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.

Access: CAT →Qπ

Input/Output:

Level 1/Argument 1		Level 1/Item 1
x	→	'$a/b*\pi$'
x	→	'a/b'
'$symb_1$'	→	'$symb_2$'
(x,y)	→	'$a/b*\pi + c/d*\pi*i$'
(x,y)	→	'$a/b + c/d*i$'

See also: →Q, /, π

QR

Type: Command

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

QR factors an $m \times n$ 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 $A \times P = Q \times R$.

Access: ↰ MATRICES FACTORIZATION QR

↰ MTH MATRIX FACTORS QR

Input/Output:

Level 1/Argument 1		Level 3/Item 1	Level 2/Item 2	Level 1/Item 3
$[[\ matrix\ ]]_A$	→	$[[\ matrix\ ]]_Q$	$[[\ matrix\ ]]_R$	$[[\ matrix\ ]]_P$

See also: LQ, LSQ

QUAD

Type: Command

Description: Solve Quadratic Equation Command: This command is identical to the computer algebra command SOLVE, and is included for backward compatibility with the HP48G.

See volume 1, CAS Commands.

Access: CAT QUAD

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
'$symb_1$'	'*global*'	→	'$symb_2$'

See also: COLCT, EXPAN, ISOL, SHOW, SOLVE

QUOTE

Type: Function

Description: Quote Argument Function: Returns its argument unevaluated.

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 placed on the stack as the argument for SIN.

This process creates a problem for functions that require symbolic arguments. For example, the integration function 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 the integral, rather than the name itself. To avoid this problem, the HP 49 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, quote the arguments using QUOTE.

Access: CAT QUOTE

Input/Output:

Level 1/Argument 1		Level 1/Item 1
obj	→	*obj*

See also: APPLY, | (Where)

RAD

Type: Command

Description: Radians Mode Command: Sets Radians angle mode.

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.

Access: CAT RAD

Input: None

Output: None

See also: DEG, GRAD

RAND

Type: Command

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

The HP 49 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.)

Access: ← MTH PROBABILITY RAND

Input/Output:

Level 1/Argument 1		Level 1/Item 1
	→	x_{random}

See also: COMB, PERM RDZ, !

RANK

Type: Command

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

Rank is computed by calculating the singular values of the matrix and counting the number of non-negligible 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.

Access: ↰ MATRICES OPERATIONS RANK

↰ MTH MATRIX NORMALIZE RANK

Input/Output:

Level 1/Argument 1		Level 1/Item 1
[[*matrix*]]	→	N_{rank}

See also: LQ, LSQ, QR

RANM

Type: Command

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

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

Access: ↰ MATRICES CREATE RANM

↰ MTH MATRIX MAKE RANM

Input/Output:

Level 1/Argument 1		Level 1/Item 1
{ *m n* }	→	[[*random matrix*]]$_{\text{m×n}}$
[[*matrix*]]$_{\text{m×n}}$	→	[[*random matrix*]]$_{\text{m×n}}$

See also: RAND, RDZ

RATIO

Type: Function

Description: Prefix Divide Function: Prefix form of / (divide).

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.

Access: CAT RATIO

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
$\tilde{z}_1$	$\tilde{z}_2$	→	$\tilde{z}_1/\tilde{z}_2$
[*array*]	{[*matrix*]}	→	[[*array* × *matrix*$^{-1}$]]
[*array*]	$\tilde{z}$	→	[*array*/$\tilde{z}$]
$\tilde{z}$	'*symb*'	→	'$\tilde{z}$/*symb*'
'*symb*'	$\tilde{z}$	→	'*symb*/$\tilde{z}$'
'$symb_1$'	'$symb_2$'	→	'$symb_1$/$symb_2$'
#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*'

RCEQ

Type: Command

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

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

Access: CAT RCEQ

Input/Output:

Level 1/Argument 1		Level 1/Item 1
	→	obj_{EQ}

See also: STEQ

RCI

Type: Command

Description: 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.

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

Access: ↰ MATRICES CREATE ROW RCI

↰ MTH MATRIX ROW RCI

Input/Output:

Level 3/Argument 1	Level 2/Argument 2	Level 1/Argument 3		Level 1/Item 1
$[[\ matrix\ ]]_1$	x_{factor}	$n_{row\ number}$	→	$[[\ matrix\ ]]_3$
$[\ vector\ ]_1$	x_{factor}	$n_{element\ number}$	→	$[\ vector\ ]_2$

See also: RCIJ

RCIJ

Type: Command

Description: 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.

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

Access: ↰ MATRICES CREATE ROW RCIJ

↰ MTH MATRIX ROW RCIJ

Input/Output:

Level 4/Argument 1	Level 3/Argument 2	Level 2/Argument 3	Level 1/Argument 4		Level 1/Item 1
$[[\ matrix\ ]]_1$	$\mathcal{X}_{\text{factor}}$	$n_{\text{row i}}$	$n_{\text{row j}}$	$\rightarrow$	$[[\ matrix\ ]]_2$
$[\ vector\ ]_1$	$\mathcal{X}_{\text{factor}}$	$n_{\text{element i}}$	$n_{\text{element j}}$	$\rightarrow$	$[\ vector\ ]_2$

See also: RCI

RCL

Type: Command Operation

Description: Recall Command: Returns the unevaluated contents of a specified variable.

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 2. 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 49 will search (in order) ports 0 through 2, and the main memory for the named backup object.

You can specify a port (that is, n_{port}) in one of three ways:

- H, 0, 1, or 2
- H, R, E, or F
- HOME, RAM, ERAM, IRAM

In each case, the ports are home, RAM, extended RAM, and Flash, respectively.

Access: ↰ RCL

Input/Output:

Level 1/Argument 1		Level 1/Item 1
'*obj*'	→	*obj*
PICT	→	*grob*
:n_{port} :$n_{library}$	→	*obj*
:n_{port} :$name_{backup}$	→	*obj*
:n_{port} :{ *path* }	→	*obj*

See also: STO

RCLALARM

Type: Command

Description: Recall Alarm Command: Recalls a specified alarm.

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

$\varkappa_{repeat}$ 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.

Access: ↱ TIME TOOLS ALRM RCLALARM

Input/Output:

Level 1/Argument 1		Level 1/Item 1
n_{index}	→	{ *date time* obj_{action} $\varkappa_{repeat}$ }

See also: DELALARM, FINDALARM, STOALARM

RCLF

Type: Command

Description: Recall Flags Command: Returns a list of integers representing the states of the system and user flags, respectively.

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.

Access: CAT RCLF

Level 1/Argument 1		Level 1/Item 1
	→	{ #n_{system} #n_{user} #$n_{system2}$ #n_{user2} }

See also: STOF

RCLKEYS

Type: Command

Description: 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.

The argument x_{key} is a real number of the form *r.c* 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.)

Access: CAT RCLKEYS

Input/Output:

Level 1/Argument 1		Level 1/Item 1
	→	{ obj_1, $x_{key\ 1}$, ... ,objn, $x_{key\ n}$ }
	→	{ S, obj_1, $x_{key\ 1}$, ... objn, $x_{key\ n}$ }

See also: ASN, DELKEYS, STOKEYS

RCLMENU

Type: Command

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

x_{menu} has the form *mm.pp*, where *mm* is the menu number and *pp* is the page of the menu.

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

Access: CAT RCLMENU

Input/Output:

Level 1/Argument 1		Level 1/Item 1
	→	x_{menu}

See also: MENU, TMENU

RCLΣ

Type: Command

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

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

Access: CAT RCLΣ

Input/Output:

Level 1/Argument 1		Level 1/Item 1
	→	*obj*

See also: CLΣ, STOΣ, Σ+, Σ–

RCWS

Type: Command

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

Access: ↱ BASE RCWS

Input/Output:

Level 1/Argument 1		Level 1/Item 1
	→	*n*

See also: BIN, DEC, HEX, OCT, STWS

RDM

Type: Command

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

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.

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

Access: ↰ MATRICES CREATE RDM

↰ MTH MATRIX MAKE RDM

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 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} }	→	

See also: TRN

RDZ

Type: Command

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

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

Access: ↰ MTH PROBABILITY RDZ

Input/Output:

Level 1/Argument 1		Level 1/Item 1
x_{seed}	→	

See also: COMB, PERM, RAND, !

RE

Type: Function

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

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.

Access: ↱ CMPLX RE

Input/Output:

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

See also: C→R, IM, R→C

RECN

Type: Command

Description: Receive Renamed Object Command: Prepares the HP 49 to receive a file from another Kermit server device, and to store the file in a specified variable.

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

Access: CAT RECN

Input/Output:

Level 1/Argument 1		Level 1/Item 1
'*name*'	→	
"*name*"	→	

See also: BAUD, CKSM, CLOSEIO, FINISH, KERRM, KGET, PARITY, RECV, SEND, SERVER, TRANSIO

RECT

Type: Command

Description: Rectangular Mode Command: Sets Rectangular coordinate mode.

RECT clears flags –15 and –16.

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

Access: CAT RECT

Input: None

Output: None

See also: CYLIN, SPHERE

RECV

Type: Command

Description: Receive Object Command: Instructs the HP 49 to look for a named file from another Kermit server device. The received file is stored in a variable named by the sender.

Since the HP 49 does not normally look for incoming Kermit files, you must use RECV to tell it to do so.

Access: CAT RECV

Input: None

Output: None

See also: BAUD, CKSM, FINISH, KGET, PARITY, RECN, SEND, SERVER, TRANSIO

RENAME

Type: Command

Description: Rename Object Command: Renames an object to the name that you specify.

Access: CAT RENAME

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
new 'name'	old 'name'	→	

See also: COPY

REPEAT

Type: Command

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

See the WHILE entry for more information.

Access: ← PRG BRANCH REPEAT

Input: None

Output: None

See also: END, WHILE

REPL

Type: Command

Description: Replace Command: Replaces a portion of the target object (first input) with a specified object (third input), beginning at a specified position (second input).

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.

Access: ⮌ PRG LIST REPL

Input/Output:

Level 3/Argument 1	Level 2/Argument 2	Level 1/Argument 3		Level 1/Item 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*$_1$	→	

See also: CHR, GOR, GXOR, NUM, POS, SIZE, SUB

RES

Type: Command

Description: 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.

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

Plot Type	Default Interval (Continued)
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 π/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)

Access: CAT RES

Input/Output:

Level 1/Argument 1		Level 1/Item 1
n_{interval}	→	
$\#n_{\text{interval}}$	→	

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

RESTORE

Type: Command

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

The specified port number must be in the range 0 to 2.

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

Access: CAT RESTORE

Input/Output:

Level 1/Argument 1		Level 1/Item 1
:n_{port} :$name_{backup}$	→	
backup	→	

See also: ARCHIVE

REVLIST

Type: Command

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

Access: ↰ PRG LIST PROCEDURES REVLIST

Input/Output:

Level 1/Argument 1		Level 1/Item 1
{ obj_n ... obj_1 }	→	{ obj_1 ... obj_n }

See also: SORT

RKF

Type: Command

Description: 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.

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}.

Access: CAT RKF

Input/Output:

L_3/A_1	L_2/A_2	L_1/A_3		L_2/I_1	L_1/I_2
{ *list* }	x_{tol}	$x_{T\ final}$	→	{ *list* }	x_{tol}
{ *list* }	{ x_{tol} x_{hstep} }	$x_{T\ final}$	→	{ *list* }	x_{tol}

L = level; A = argument; I = item

See also: RKFERR, RKFSTEP, RRK, RRKSTEP, RSBERR

RKFERR

Type: Command

Description: 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.

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.)

Access: CAT RKFE

Input/Output:

L_2/A_1	L_1/A_2		L_4/I_1	L_3/I_2	L_2/I_3	L_1/I_4
{ *list* }	*h*	→	{ *list* }	*h*	y_{delta}	*error*

L = level; A = argument; I = item

See also: RKF, RKFSTEP, RRK, RRKSTEP, RSBERR

RKFSTEP

Type: Command

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

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 methods 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.

Access: CAT RKFS

Input/Output:

L_3/A_1	L_2/A_n	L_1/A_{n+1}		L_3/I_1	L_2/I_2	L_1/I_3
{ *list* }	x_{tol}	h	→	{ *list* }	x_{tol}	h_{next}

L = level; A = argument; I = item

See also: RKF, RKFERR, RRK, RRKSTEP, RSBERR

RL

Type: Command

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

The leftmost bit of $\#n_1$ becomes the rightmost bit of $\#n_2$.

Access: ← MTH BASE BIT RL

→ BASE BIT RL

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\#n_1$	→	$\#n_2$

See also: RLB, RR, RRB

RLB

Type: Command

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

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

Access: ↰ MTH BASE BYTE RLB

↱ BASE BYTE RLB

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\#n_1$	→	$\#n_2$

See also: RL, RR, RRB

RND

Type: Function

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

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

n_{round} or $symb_{\text{round}}$	Effect on Level 2 Argument
0 through 1	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.

Access: ↰ MTH REAL RND

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
$\tilde{z}_1$	n_{round}	→	$\tilde{z}_2$
$\tilde{z}$	'$symb_{round}$'	→	'RND($symb_{round}$)'
'$symb$'	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, $symb_{round}$)'

See also: TRNC

RNRM

Type: Command

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

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.

Access: ↰ MATRICES OPERATIONS RNRM

↰ MTH MATRIX NORMALIZE RNRM

Input/Output:

Level 1/Argument 1		Level 1/Item 1
[$array$]	→	$x_{row\ norm}$

See also: CNRM, CROSS, DET, DOT

ROLL

Type: RPL command

Description: 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.

In RPN mode, 3 ROLL is equivalent to ROT.

Access: ↰ PRG STACK ROLL

Input/Output:

L_{n+1}... L_2	L_1		L_n ...	L_2	L_1
obj_n ... obj_1	n	→	obj_{n-1} ...	obj_1	obj_n

L = level

See also: OVER, PICK, ROLLD, ROT, SWAP

ROLLD

Type: RPL command

Description: Roll Down Command: Moves the contents of level 2 to a specified level, *n*, and rolls downward the portion of the stack beneath the specified level.

Access: ↰ PRG STACK ROLLD

Input/Output:

L_{n+1}... L_2	L_1		L_n	L_{n-1} ...	L_1
obj_n ... obj_2	n (obj_1)	→	obj_1	obj_n ...	obj_2

L = level

See also: OVER, PICK, ROLL, ROT, SWAP

ROMUPLOAD

Type: Command

Description: Upload the calculator's operating system to another calculator.. Use this command to upload your ROM to another calculator.

1. On the sending calculator, enter the ROMUPLOAD command and press ENTER. The calculator displays a screen with instructions on how to download the ROM.
2. On the receiving calculator, hold down the ON and press the F4. The calculator displays the Test menu.
3. On the receiving calculator, hold down ON and + and press ENTER. After a short time, the Terminal menu appears.
4. On the receiving calculator, press 4 to select the Download option.
5. On the sending calculator, press any key to start the download process. The process takes around 20 minutes.

Access: CAT

ROOT

Type: Command

Description: 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.

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.

Access: CAT ROOT

Input/Output:

Level 3/Argument 1	Level 2/Argument 2	Level 1/Argument 3		Level 1/Item 1
«*program*»	'*global*'	*guess*	→	x_{root}
«*program*»	'*global*'	{ *guesses* }	→	x_{root}
'*symb*'	'*global*'	*guess*	→	x_{root}
'*symb*'	'*global*'	{ *guesses* }	→	x_{root}

ROT

Type: RPL ommand

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

In RPN mode, ROT is equivalent to 3 ROLL.

Access: ↰ PRG STACK ROT

Input/Output:

L_3	L_2	L_1		L_3	L_2	L_1
obj_3	obj_2	obj_1	→	obj_2	obj_1	obj_3

L = level

See also: OVER, PICK, ROLL, ROLLD, SWAP, UNROT

ROW–

Type: Command

Description: 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).

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

Access: ↰ MATRICES CRREATE ROW ROW–

↰ MTH MATRIX ROW ROW–

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 2/Item 1	Level 1/Item 2
$[[\ matrix\ ]]_1$	n_{row}	→	$[[\ matrix\ ]]_2$	$[\ vector\ ]_{row}$
$[\ vector\ ]_1$	$n_{element}$	→	$[\ vector\ ]_2$	$element_n$

See also: COL–, COL+, ROW+, RSWP

ROW+

Type: Command

Description: 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).

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.

Access: **Access:** ↰ MATRICES CRREATE ROW ROW+

↰ MTH MATRIX ROW ROW+

Input/Output:

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

See also: COL–, COL+, ROW–, RSWP

ROW→

Type: Command

Description: 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.

Access: MATR ROW ROW→

Input/Output:

L_{n+1}/A_1 ...	L_2/A_n	L_1/A_{n+1}		Level 1/Item 1
$[\ vector\ ]_{row\ 1}$ ...	$[\ vector\ ]_{row\ n}$	$n_{row\ count}$	→	$[[\ matrix\ ]]$
$element_1$ ...	$element_n$	$n_{element\ count}$	→	$[\ vector\ ]_{column}$

L = level; A = argument; I = item

See also: →COL, COL→, →ROW

→ROW

Type: Command

Description: 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.

Access: ↰ MATRICES CREATE ROW →ROW

↰ MTH MATRIX ROW →ROW

Input/Output:

L_1/Argument$_1$		L_{n+1}/I_1 … L_2/I_n	L_1/I_{n+1}
[[*matrix*]]	→	[*vector*]$_{\text{row n}}$ … [*vector*]$_{\text{row n}}$	n_{rowcount}
[*vector*]	→	$element_1$ … $element_n$	$n_{\text{elementcount}}$

L =Level; A = Argument; I = Item

See also: →COL, COL→, ROW→

RR

Type: Command

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

The rightmost bit of $\#n_1$ becomes the leftmost bit of $\#n_2$.

Access: ↱ BASE BIT RR

↰ MTH BASE BIT RR

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\#n_1$	→	$\#n_2$

See also: RL, RLB, RRB

RRB

Type: Command

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

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

Access: ↱ BASE BYTE RRB

↰ MTH BASE BYTE RRB

Input/Output:

Level 1/Argument 1		Level 1/Item 1
#n_1	→	#n_2

See also: RL, RLB, RR

RRK

Type: Command

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

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 its steps from the initial value to x_{Tfinal}.

Access: CAT RRK

Input/Output:

L_3/A_1	L_2/A_2	L_1/A_3		L_2/I_1	L_1/I_2
{ *list* }	x_{tol}	$x_{T\ final}$	→	{ *list* }	x_{tol}
{ *list* }	{ x_{tol} x_{hstep} }	$x_{T\ final}$		{ *list* }	x_{tol}

L = level; A = argument; I = item

See also: RKF, RKFERR, RKFSTEP, RRKSTEP, RSBERR

RRKSTEP

Type: Command

Description: 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.

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.

Access: CAT RRKS

Input/Output:

L_4/A_1	L_3/A_2	L_2/A_3	L_1/A_4		L_4/I_1	L_3/I_2	L_2/I_3	L_1/I_4
{ *list* }	x_{tol}	*h*	*last*	→	{ *list* }	x_{tol}	h_{next}	*current*

L = level; A = argument; I = item

See also: RKF, RKFERR, RKFSTEP, RRK, RSBERR

RSBERR

Type: Command

Description: 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.

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.

Access: CAT RSBER

Input/Output:

L_2/A_1	L_1/A_2		L_4/I_1	L_3/I_2	L_2/I_3	L_1/I_4
{ *list* }	*h*	→	{ *list* }	*h*	y_{delta}	*error*

L = level; A = argument; I = item

See also: RKF, RKFERR, RKFSTEP, RRK, RRKSTEP

RSD

Type: Command

Description: Residual Command: Computes the residual B – AZ of the arrays B, A, and Z.

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 AX = B.

Access: ↰ MATRICES OPERATIONS RSD

↰ MTH MATRIX RSD

Input/Output:

Level 3/Argument 1	Level 2/Argument 2	Level 1/Argument 3		Level 1/Item 1
$[\ vector\ ]_B$	$[[\ matrix\ ]]_A$	$[\ vector\ ]_Z$	→	$[\ vector\ ]_{B-AZ}$
$[[\ matrix\ ]]_B$	$[[\ matrix\ ]]_A$	$[[\ matrix\ ]]_Z$	→	$[[\ matrix\ ]]_{B-AZ}$

RSWP

Type: Command

Description: 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.

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

Access: ↰ MATRICES CREATE ROW RSWP

↰ MTH MATRIX ROW RSWP

Input/Output:

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

See also: CSWP, ROW+, ROW–

R→B

Type: Command

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

For any value of $n \leq 0$, the result is # 0. For any value of $n \geq 1.84467440738E19$ (base 10), the result is # FFFFFFFFFFFFFFFF (base 16).

Access: ↱ BASE R→B

Input/Output:

Level 1/Argument 1		Level 1/Item 1
n	→	#*n*

See also: B→R

R→C

Type: Command

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

The firts input represents the real element(s) of the complex result. The second input represents the imaginary element(s) of the complex result.

Array arguments must have the same dimensions.

Access: ↰ PRG TYPE R→C

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
x	*y*	→	(*x*,*y*)
[*R-array*$_1$]	[*R-array*$_2$]	→	[*C-array*]

See also: C→R, IM, RE

R→D

Type: Function

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

This function operates independently of the angle mode.

Access: ↰ MTH REAL R→D

Input/Output:

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

FSee also: D→R

S

SAME

Type: Command

Description: 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.

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.

Access: ↰ PRG TEST SAME

Input/Output:

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

See also: TYPE, ==

SBRK

Type: Command

Description: Serial Break Command: Interrupts serial transmission or reception.

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

Access: CAT SBRK

Input: None

Output: None

See also: BUFLEN, SRECV, STIME, XMIT

SCALE

Type: Command

Description: 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.

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.

Access: CAT SCALE

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
x_{scale}	y_{scale}	→	

See also: AUTO, CENTR, SCALEH, SCALEW

SCALEH

Type: Command

Description: Multiply Height Command: Multiplies the vertical plot scale by x_{factor}.

Executing SCALEH changes the *y*-axis display range—the y_{min} and y_{max} components of the first two complex numbers in the reserved variable PPAR. The plot origin (the user-unit coordinate of the center pixel) is not changed.

Access: CAT SCALEH

Input/Output:

Level 1/Argument 1		Level 1/Item 1
x_{factor}	→	

See also: AUTO, SCALEW, YRING

SCALEW

Type: Command

Description: Multiply Width Command: Multiplies the horizontal plot scale by x_{factor}.

Executing SCALEH changes the x-axis display range—the x_{min} and x_{max} components of the first two complex numbers in the reserved variable PPAR. The plot origin (the user-unit coordinate of the center pixel) is not changed.

Access: CAT SCALEW

Input/Output:

Level 1/Argument 1		Level 1/Item 1
x_{factor}	→	

See also: AUTO, SCALEH, YRING

SCATRPLOT

Type: Command

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

The data columns plotted are specified by XCOL and YCOL, and are stored as the first two parameters in the reserved variable Σ*PAR*. 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.

Access: CAT SCATTERPLOT

Input: None

Output: None

See also: BARPLOT, PICTURE, HISTPLOT, PVIEW, SCLΣ, XCOL, YCOL

SCATTER

Type: Command

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

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:

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

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,–3.1).
- (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. 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*.

Access: CAT SCATTER

Input: None

Output: None

See also: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

SCHUR

Type: Command

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

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, $A \cong Q \times T \times \text{TRN}(Q)$

Access: ↰ MATRICES FACTORIZATION SCHUR

↰ MTH MATRIX FACTORS SCHUR

Input/Output:

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

See also: LQ, LU, QR, SVD, SVL, TRN

SCI

Type: Command

Description: 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.

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.

Access: CAT SCI

Input/Output:

Level 1/Argument 1		Level 1/Item 1
n	→	

See also: ENG, FIX, STD

SCLΣ

Type: Command

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

When the plot type is SCATTER, the command AUTO incorporates the functions of SCLΣ. In addition, the command SCATTERPLOT automatically executes AUTO to achieve the same result.

Access: CAT SCLΣ

Input: None

Output: None

See also: AUTO, SCATRPLOT

SCONJ

Type: Command

Description: Store Conjugate Command: Conjugates the contents of a named object.

The named object must be a number, an array, or an algebraic object. For information on conjugation, see CONJ.

Access: ↰ PRG MEMORY ARITHMETIC SCONJ

Input/Output:

Level 1/Argument 1		Level 1/Item 1
'*name*'	→	

See also: CONJ, SINV, SNEG

SCROLL

Type: Command

Description: .

Access: CAT

Input/Output:

Level 1/Argument 1		Level 1/Item 1
	→	

See also:

SDEV

Type: Command

Description: Standard Deviation Command: Calculates the sample standard deviation of each of the *m* columns of coordinate values in the current statistics matrix (reserved variable Σ*DAT*).

SDEV returns a vector of *m* real numbers, or a single real number if *m* = 1. The standard deviation (the square root of the variances) is computed using this formula:

$$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

where x_i is the *i*th coordinate value in a column, $\bar{x}$ is the mean of the data in this column, and *n* is the number of data points.

Access: CAT SDEV

Input/Output:

Level 1/Argument 1		Level 1/Item 1
	→	x_{sdev}
	→	$[\, x_{sdev\,1}\; x_{sdev\,2} \ldots x_{sdev\,m} \,]$

See also: MAXΣ, MEAN, MINΣ, PSDEV, PVAR, TOT, VAR

SEND

Type: Command

Description: Send Object Command: Sends a copy of the named objects to a Kermit device.

Data is always sent from a local Kermit, but can be sent either to another local Kermit (which must execute RECV or RECN) or to a server Kermit.

To rename an object when sending it, include the old and new names in an embedded list.

Access: CAT SEND

Input/Output:

Level 1/Argument 1		Level 1/Item 1
'*name*'	→	
{ $name_1$ ... $name_n$ }	→	
{{ $name_{old}$ $name_{new}$ } *name* ... }	→	

See also: BAUD, CLOSEIO, CKSM, FINISH, KERRM, KGET, PARITY, RECN, RECV, SERVER, TRANSIO

SEQ

Type: Command

Description: Sequential Calculation Command: Returns a list of results generated by repeatedly executing obj_{exec} using *index* over the range x_{start} to x_{end}, in increments of x_{incr}.

obj_{exec} is nominally a program or algebraic object that is a function of *index*, but can actually be an object. *index* must be a global or local name. The remaining objects can be anything that will evaluate to real numbers.

The action of SEQ for arbitrary inputs can be predicted exactly from this equivalent program.

$$x_{start}\ x_{end}\ \text{FOR}\ index\ obj_{exec}\ \text{EVAL}\ x_{incr}\ \text{STEP}\ n \rightarrow \text{LIST}$$

where *n* is the number of new objects left on the stack by the FOR … STEP loop. Notice that *index* becomes a local variable regardless of its original type.

Access: ↰ PRG LIST PROCEDURES SEQ

Input/Output:

L_5/A_1	L_4/A_2	L_3/A_3	L_2/A_4	L_1/A_5		L_1/I_1
obj_{exec}	*index*	x_{start}	x_{end}	x_{incr}	→	{ *list* }

L = level; A = argument; I = item

FSee also: DOSUBS, STREAM

SERVER

Type: Command

Description: Server Mode Command: Selects Kermit Server mode.

A Kermit server (a Kermit device in Server mode) passively processes requests sent to it by the local Kermit. The server receives data in response to SEND, transmits data in response to KGET, terminates Server mode in response to FINISH or LOGOUT, and transmits a directory listing in response to a generic directory request.

Access: CAT SERVER

Input: None

Output: None

See also: BAUD, CKSM, FINISH, KERRM, KGET, PARITY, PKT, RECN, RECV, SEND, TRANSIO

SF

Type: Command

Description: Set Flag Command: Sets a specified user or system flag.

User flags are numbered 1 through 128. System flags are numbered –1 through –128. See the HP 49G *Pocket Guide* for a listing of system flags and their flag numbers.

Access: ⮌ PRG TEST SF

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$n_{\text{flagnumber}}$	→	

See also: CF, FC?, FC?C, FS?, FS?C

SHOW

Type: Command

Description: Show Variable Command: Returns $symb_2$, which is equivalent to $symb_1$ except that all implicit references to a variable *name* are made explicit.

If the level 1 argument is a list, SHOW evaluates all global variables in $symb_1$ *not* contained in the list.

Access: CAT SHOW

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
'$symb_1$'	'*name*'	→	'$symb_2$'
'$symb_1$'	{ $name_1$ $name_2$ ... }	→	'$symb_2$'

See also: COLCT, EXPAN, ISOL, QUAD

SIDENS

Type: Function

Description: Silicon Intrinsic Density Command: Calculates the intrinsic density of silicon as a function of temperature, x_T.

If x_T is a unit object, it must reduce to a pure temperature, and the density is returned as a unit object with units of $1/cm^3$.

If x_T is a real number, its units are assumed to be K, and the density is returned as a real number with implied units of $1/cm^3$.

x_T must be between 0 and 1685 K.

Access: CAT SIDEN

Input/Output:

Level 1/Argument 1	→	Level 1/Item 1
x_T		$x_{density}$
x_unit		x_1/cm^3
'*symb*'		'SIDENS(*symb*)'

SIGN

Type: Function

Description: Sign Function: Returns the sign of a real number argument, the sign of the numerical part of a unit object argument, or the unit vector in the direction of a complex number argument.

For real number and unit object arguments, the sign is defined as +1 for positive arguments, –1 for negative arguments, In exact mode, the sign for argument 0 is undefined (?). In approximate mode, the sign for argument 0 is 0.

For a complex argument:

$$\text{SIGN}(x+iy) = \frac{x}{\sqrt{x^2+y^2}} + \frac{iy}{\sqrt{x^2+y^2}}$$

Access: ↰ MTH REAL SIGN (returns the sign of a number)

↱ CMPLX SIGN (returns the unit vector of a complex number)

Input/Output:

Level 1/Argument 1	→	Level 1/Item 1
$\tilde{z}_1$		$\tilde{z}_2$
x_unit		x_{sign}
'*symb*'		'SIGN(*symb*)'

See also: ABS, MANT, XPON

SIN

Type: Analytic function

Description: Sine Analytic Function: Returns the sine of the argument.

For real arguments, the current angle mode determines the number's units, unless angular units are specified.

For complex arguments, $\sin(x + iy) = \sin x \cosh y + i \cos x \sinh y$.

If the argument for SIN is a unit object, then the specified angular unit overrides the angle mode to determine the result. Integration and differentiation, on the other hand, always observe the angle mode. Therefore, to correctly integrate or differentiate expressions containing SIN with a unit object, the angle mode must be set to radians (since this is a "neutral" mode).

Access: SIN

Input/Output:

Level 1/Argument 1	→	Level 1/Item 1
$\tilde{z}$		$\sin \tilde{z}$
$x\_unit_{angular}$		$\sin(x\_unit_{angular})$
'*symb*'		'SIN(*symb*)'

See also: ASIN, COS, TAN

SINH

Type: Analytic function

Description: Hyperbolic Sine Analytic Function: Returns the hyperbolic sine of the argument.

For complex arguments, $\sinh(x + iy) = \sinh x \cos y + i \cosh x \sin y$.

Access: ↱ TRIG HYPERBOLIC SINH

↰ MTH HYPERBOLIC SINH

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\tilde{z}$	→	sinh $\tilde{z}$
'*symb*'		'SINH(*symb*)'

See also: ASINH, COSH, TANH

SINV

Type: Command

Description: Store Inverse Command: Replaces the contents of the named variable with its inverse.

The named object must be a number, a matrix, an algebraic object, or a unit object. For information on reciprocals, see INV.

Access: ↰ PRG MEMORY ARITHMETIC SINV

Input/Output:

Level 1/Argument 1		Level 1/Item 1
'*name*'	→	

See also: INV, SCONJ, SNEG

SIZE

Type: Command Operation

Description: Size Command: Returns the number of characters in a string, the number of elements in a list, the dimensions of an array, the number of objects in a unit object or algebraic object, or the dimensions of a graphics object.

The size of a unit is computed as follows: the scalar (+1), the underscore (+1), each unit name (+1), operator or exponent (+1), and each prefix (+2).

Any object type not listed above returns a value of 1.

Access: ↰ PRG CHARS SIZE

Input/Output:

Level 1/Argument 1		Level 2/Item 1	Level 1/Item 2
"*string*"	→		*n*
{ *list* }	→		*n*
[*vector*]	→		{ *n* }
[[*matrix*]]	→		{ *n m*}
'*symb*'	→		*n*
grob	→	$\#n_{\text{width}}$	$\#m_{\text{height}}$
PICT	→	$\#n_{\text{width}}$	$\#m_{\text{height}}$
x_unit	→		*n*

See also: CHR, NUM, POS, REPL, SUB

SL

Type: Command

Description: Shift Left Command: Shift a binary integer one bit to the left.

The most significant bit is shifted out to the left and lost, while the least significant bit is regenerated as a zero. SL is equivalent to binary multiplication by 2, truncated to the current wordsize.

Access: ↰ MTH BASE BIT SL

↱ BASE BIT SL

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\#n_1$	→	$\#n_2$

See also: ASR, SLB, SR, SRB

SLB

Type: Command

Description: Shift Left Byte Command: Shifts a binary integer one byte to the left.

The most significant byte is shifted out to the left and lost, while the least significant byte is regenerated as zero. SLB is equivalent to binary multiplication by 28 (or executing SL eight times), truncated to the current wordsize.

Access: ↰ MTH BASE BYTE SLB

↱ BASE BYTE SLB

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\#n_1$	→	$\#n_2$

See also: ASR, SL, SR, SRB

SLOPEFIELD

Type: Command

Description: SLOPEFIELD Plot Type Command: Sets the plot type to SLOPEFIELD.

When plot type is set to SLOPEFIELD, the DRAW command plots a slope representation of a scalar function with two variables. SLOPEFIELD requires values in the reserved variables *EQ*, *VPAR*, and *PPAR*.

VPAR has the following form:

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

For plot type SLOPEFIELD, 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:

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

For plot type SLOPEFIELD, 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 SLOPEFIELD places the command name SLOPEFIELD in *ptype*.
- *depend* is a name specifying the dependent variable. The default value is *Y*.

Access: CAT SLOPEFIELD

Input: None

Output: None

See also: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, TRUTH, WIREFRAME, YSLICE

SNEG

Type: Command

Description: Store Negate Command: Replaces the contents of a variable with its negative.

The named object must be a number, an array, an algebraic object, a unit object, or a graphics object. For information on negation, see NEG.

Access: ↰ PRG MEMORY ARITHMETIC SNEG

Input/Output:

Level 1/Argument 1		Level 1/Item 1
'*name*'	→	

See also: NEG, SCONJ, SINV

SNRM

Type: Command

Description: Spectral Norm Command: Returns the spectral norm of an array.

The spectral norm of a vector is its Euclidean length, and is equal to the largest singular value of a matrix.

Access: ↰ MATRICES OPERATIONS SNRM

↰ MTH MATRIX NORMALIZE SNRM

Input/Output:

Level 1/Argument 1		Level 1/Item 1
[*array*]	→	$x_{\text{spectralnorm}}$

See also: ABS, CNRM, COND, RNRM, SRAD, TRACE

SOLVER

Type: Command

.Description: Displays a menu of commands used in solving equatons.

Access: CAT SOLVER

Input: None

Output: None

SORT

Type: Command

Description: Ascending Order Sort Command: Sorts the elements in a list in ascending order.

The elements in the list can be real numbers, strings, lists, names, binary integers, or unit objects. However, all elements in the list must all be of the same type. Strings and names are sorted by character code number. Lists of lists are sorted by the first element in each list.

To sort in reverse order, use SORT REVLIST.

Access: ↰ MTH LIST SORT

↰ PRG LIST PROCEDURES SORT

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\{ list \}_1$	→	$\{ list \}_2$

See also: REVLIST

SPHERE

Type: Command

Description: Spherical Mode Command: Sets spherical coordinate mode.

SPHERE sets flags –15 and –16.

In spherical mode, vectors are displayed as polar components.

Access: CAT SPHERE

Input: None

Output: None

See also: CYLIN, RECT

SQ

Type: Analytic function

Description: Square Analytic Function: Returns the square of the argument.

The square of a complex argument (x, y) is the complex number $(x^2 - y^2, 2xy)$.

Matrix arguments must be square.

Access: ↰ x^2

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\tilde{z}$	→	$\tilde{z}^2$
x_unit	→	x^2*_unit*^{2}
[[*matrix*]]	→	[[*matrix* × *matrix*]]
'*symb*'	→	'SQ(*symb*)'

See also: √, ^

SR

Type: Command

Description: Shift Right Command: Shifts a binary integer one bit to the right.

The least significant bit is shifted out to the right and lost, while the most significant bit is regenerated as a zero. SR is equivalent to binary division by 2.

Access: ↰ MTH BASE BIT SR

↱ BASE BIT SR

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\#n_1$	→	$\#n_2$

See also: ASR, SL, SLB, SRB

SRAD

Type: Command

Description: Spectral Radius Command: Returns the spectral radius of a square matrix.

The spectral radius of a matrix is a measure of the size of the matrix, and is equal to the absolute value of the largest eigenvalue of the matrix.

Access: ↰ MATRICES OPERATIONS SRAD

↰ MTH MATRIX NORMALIZE

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$[[\ matrix\ ]]_{n\times n}$	→	$\varkappa_{\text{spectralradius}}$

See also: COND, SNRM, TRACE

SRB

Type: Command

Description: Shift Right Byte Command: Shifts a binary integer one byte to the right.

The least significant byte is shifted out to the right and lost, while the most significant byte is regenerated as zero. SRB is equivalent to binary division by 2^8 (or executing SR eight times).

Access: ↰ MTH BASE BYTE SRB

↱ BASE BYTE SRB

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\#n_1$	→	$\#n_2$

See also: ASR, SL, SLB, SR

SRECV

Type: Command

Description: Serial Receive Command: Reads up to *n* characters from the serial input buffer and returns them as a string, along with a digit indicating whether errors occurred.

SRECV does not use Kermit protocol.

If *n* characters are not received within the time specified by STIME (default is 10 seconds), SRECV "times out", returning a 0 to level 1 and as many characters as were received to level 2.

If the level 2 output from BUFLEN is used as the input for SRECV, SRECV will not have to wait for more characters to be received. Instead, it returns the characters already in the input buffer.

If you want to accumulate bytes in the input buffer before executing SRECV, you must first open the port using OPENIO (if the port isn't already open).

SRECV can detect three types of error when reading the input buffer:

- Framing errors and UART overruns (both causing "Receive Error" in ERRM).
- Input-buffer overflows (causing "Receive Buffer Overflow" in ERRM).
- Parity errors (causing "Parity Error" in ERRM).

SRECV returns 0 if an error is detected when reading the input buffer, or 1 if no error is detected.

Parity errors do not stop data flow into the input buffer. However, if a parity error occurs, SRECV stops reading data after encountering a character with an error.

Framing, overrun, and overflow errors cause all subsequently received characters to be ignored until the error is cleared. SRECV does not detect and clear any of these types of errors until it tries to read the byte where the error occurred. Since these three errors cause the byte where the error occurred and all subsequent bytes to be ignored, the input buffer will be empty after all previously received good bytes have been read. Therefore, SRECV detects and clears these errors when it tries to read a byte from an empty input buffer.

Note that BUFLEN also clears the above-mentioned framing, overrun, and overflow errors. Therefore, SRECV cannot detect an input-buffer overflow after BUFLEN is executed, unless more characters were received after BUFLEN was executed (causing the input buffer to overflow again). SRECV also cannot detect framing and UART overrun errors cleared by BUFLEN. To find where the data error occurred, save the number of characters returned by BUFLEN (which gives the number of "good" characters received), because as soon as the error is cleared, new characters can enter the input buffer.

Access: CAT SRECV

Input/Output:

Level 1/Argument 1		Level 2/Item 1	Level 1/Item 2
n	→	'*string*'	0/1

See also: BUFLEN, CLOSEIO, OPENIO, SBRK, STIME, XMIT

SREPL

Type: Command

Description: Find and replace: Finds and replaces a string in a given text object. You supply the following inputs:
Level 3/argument 1: the string to search.
Level 2/argument 2: the string to find.
Level 1/argument 3: the string to replace it with.

Access: CAT

Input/Output:

Level 3/Argument 1	Level 2/Argument 2	Level 1/Argument 3		Level 1/Item 1
'*string*'	'*string*'	'*string*'	→	'*string*'

START

Type: Command Operation

Description: START Definite Loop Structure Command: Begins START … NEXT and START … STEP definite loop structures.

Definite loop structures execute a command or sequence of commands a specified number of times.

- START … NEXT executes a portion of a program a specified number of times. The RPL syntax is this:

  x_{start} x_{finish} START *loop-clause* NEXT

  The algebraic syntax is this:

  START(x_{start} x_{finish}) *loop-clause* NEXT

  START takes two numbers (x_{start} and x_{finish}) from the stack and stores them as the starting and ending values for a loop counter. Then the loop clause is executed. NEXT increments the counter by 1 and tests to see if its value is less than or equal to x_{finish}. If so, the loop clause is executed again. Notice that the loop clause is always executed at least once.

- START … STEP works just like START … NEXT, except that it can use an increment value other than 1. The RPL syntax is this:

  x_{start} x_{finish} START *loop-clause* $x_{increment}$ STEP

  The algebraic syntax is this:

  START (x_{start} x_{finish}) *loop-clause* STEP($x_{increment}$)

  START takes two numbers (x_{start} and x_{finish}) from the stack and stores them as the starting and ending values for the loop counter. Then the loop clause is executed. STEP takes $x_{increment}$ from the stack and increments the counter by that value. If the argument of STEP is an algebraic or a name, it is automatically evaluated to a number.

The increment value can be positive or negative:

– If positive, the loop is executed again when the counter is less than or equal to x_{finish}.

– If negative, the loop is executed when the counter is greater than or equal to x_{finish}.

Access: ⬅ PRG BRANCH START

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
START x_{start}	x_{finish}	→	
NEXT		→	
STEP	$x_{increment}$	→	
STEP	'$symb_{increment}$'	→	

See also: FOR, NEXT, STEP

STD

Type: Command

Description: Standard Mode Command: Sets the number display format to standard mode.

Executing STD has the same effect as clearing flags –49 and –50.

Standard format produces the following results when displaying or printing a number.

- Numbers that can be represented exactly as integers with 12 or fewer digits are displayed without a fraction mark or exponent. Zero is displayed as 0.
- Numbers that can be represented exactly with 12 or fewer digits, but not as integers, are displayed with a fraction mark but no exponent. Leading zeros to the left of the fraction mark and trailing zeros to the right of the fraction mark are omitted.
- All other numbers are displayed in scientific notation (see SCI) with both a fraction mark (with one number to the left) and an exponent (of one or three digits). There are no leading or trailing zeros.

In algebraic objects, integers less than 10^3 are always displayed in standard mode.

Access: CAT STD

Input: None

Output: None

See also: ENG, FIX, SCI

STEP

Type: Command Operation

Description: STEP Command: Defines the increment (step) value, and ends definite loop structure.

See the FOR and START keyword entries for more information.

Access: ↰ PRG BRANCH STEP

Input: None

Output: None

See also: FOR, NEXT, START

STEQ

Type: Command

Description: Store in EQ Command: Stores an object into the reserved variable *EQ* in the current directory.

Access: CAT STEQ

Input/Output:

Level 1/Argument 1		Level 1/Item 1
obj	→	

See also: RCEQ

STIME

Type: Command

Description: Serial Time-Out Command: Specifies the period that SRECV (serial reception) and XMIT (serial transmission) wait before timing out.

The value for *x* is interpreted as a positive value from 0 to 25.4 seconds. If no value is given, the default is 10 seconds. If *x* is 0, there is no time-out; that is, the device waits indefinitely, which can drain the batteries.

STIME is not used for Kermit time-out.

Access: CAT STIME

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\mathcal{X}_{\text{seconds}}$	→	
0	→	

See also: BUFLEN, CLOSEIO, SBRK, SRECV, XMIT

STO

Type: Command

Description: Store Command: Stores an object into a specified variable or object.

Storing a graphics object into *PICT* makes it the current graphics object.

To create a backup object, store the *obj* into the desired backup location (identified as :n_{port}:$name_{\text{backup}}$). STO will not overwrite an existing backup object.

To store backup objects and library objects, specify a port number (0 through 2).

After storing a library object in a port, it must then be attached to its directory before it can be used. The easiest way to do this is to execute a warm start (by pressing ON F3). This also causes the calculator to perform a *system halt*, which clears the stack, the LAST stack, and all local variables.

STO can also replace a single element of an array or list stored in a variable. Specify the variable in level 1 as *name*(*index*), which is a user function with *index* as the argument. The *index* can be *n* or *n*,*m*, where *n* specifies the row position in a vector or list, and *n*,*m* specifies the row-and-column position in a matrix.

Access: STO▸

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
obj	'*name*'	→	
grob	*PICT*	→	
obj	:n_{port} :$name_{\text{backup}}$	→	
obj	'*name*(*index*)'	→	
backup	n_{port}	→	
library	n_{port}	→	
library	:n_{port} :n_{library}	→	

See also: DEFINE, RCL, →,

STO–

Type: Command

Description: Store Minus Command: Calculates the difference between a number (or other object) and the contents of a specified variable, and stores the new value in the specified variable.

The object on the stack and the object in the variable must be suitable for subtraction with each other. STO– can subtract any combination of objects suitable for stack subtraction.

Using STO– to subtract two arrays (where *obj* is an array and *name* is the global name of an array) requires less memory than using the stack to subtract them.

Access: ↰ PRG MEMORY ARITHMETIC STO–

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
obj	'*name*'	→	
'*name*'	*obj*	→	

See also: STO+, STO*, STO/, –

STO*

Type: Command

Description: Store Times Command: Multiplies the contents of a specified variable by a number or other object.

The object on the stack and the object in the variable must be suitable for multiplication with each other. When multiplying two arrays, the result depends on the order of the arguments. The new object of the named variable is the level 2 array times the level 1 array. The arrays must be conformable for multiplication.

Using STO* to multiply two arrays or to multiply a number and an array (where *obj* is an array or a number and *name* is the global name of an array) requires less memory than using the stack to multiply them.

Access: ↰ PRG MEMORY ARITHMETIC STO*

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
obj	'*name*'	→	
'*name*'	*obj*	→	

See also: STO+, STO–, STO/, *

STO/

Type: Command

Description: Store Divide Command: Calculates the quotient of a number (or other object) and the contents of a specified variable, and stores the new value in the specified variable.

The new object of the specified variable is the level 2 object divided by the level 1 object.

The object on the stack and the object in the variable must be suitable for division with each other. If both objects are arrays, the divisor (level 1) must be a square matrix, and the dividend (level 2) must have the same number of columns as the divisor.

Using STO/ to divide one array by another array or to divide an array by a number (where *obj* is an array or a number and *name* is the global name of an array) requires less memory than using the stack to divide them.

Access: ↰ PRG MEMORY ARITHMETIC STO/

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
obj	'*name*'	→	
'*name*'	*obj*	→	

See also: STO+, STO–, STO*, /

STO+

Type: Command

Description: Store Plus Command: Adds a number or other object to the contents of a specified variable.

The object on the stack and the object in the variable must be suitable for addition to each other. STO+ can add any combination of objects suitable for addition.

Using STO+ to add two arrays (where *obj* is an array and *name* is the global name of an array) requires less memory than using the stack to add them.

Access: ↰ PRG MEMORY ARITHMETIC STO+

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
obj	'*name*'	→	
'*name*'	*obj*	→	

FSee also: STO–, STO*, STO/, +

STOALARM

Type: Command

Description: Store Alarm Command: Stores an alarm in the system alarm list and returns its alarm index number.

If the argument is a real number x_{time}, the alarm date will be the current system date by default.

If obj_{action} is a string, the alarm is an appointment alarm, and the string is the alarm message. If obj_{action} is any other object type, the alarm is a control alarm, and the object is executed when the alarm comes due.

x_{repeat} is the repeat interval for the alarm in clock ticks, where 8192 ticks equals 1 second.

n_{index} is a real integer identifying the alarm based on its chronological position in the system alarm list.

Access: ▶ TIME TOOLS ALRM STOALARM

Input/Output:

Level 1/Argument 1		Level 1/Item 1
x_{time}	→	n_{index}
{ *date time* }	→	n_{index}
{ *date time obj*$_{action}$ }	→	n_{index}
{ *date time obj*$_{action}$ x_{repeat} }	→	n_{index}

See also: DELALARM, FINDALARM, RCLALARM

STOF

Type: Command

Description: Store Flags Command: Sets the states of the system flags or the system and user flags.

With argument #n_{system}, STOF sets the states of the system flags (–1 through –64) only. With argument { #n_{system}, #n_{user}, #$n_{system2}$ #n_{user2} }, STOF sets the states of both the system and user flags.

A bit with value 1 sets the corresponding flag; a bit with value 0 clears the corresponding flag. The rightmost (least significant) bit of #n_{system} and #n_{user} correspond to the states of system flag –1 and user flag +1, respectively.

STOF can preserve the states of flags before a program executes and changes the states. RCLF can then recall the flag's states after the program is executed.

Access: CAT STOF

Input/Output:

Level 1/Argument 1		Level 1/Item 1
#n_{system}	→	
{ #n_{system} #n_{user} #$n_{system2}$ #n_{user2} }	→	

See also: RCLF, STWS, RCWS

STOKEYS

Type: Command

Description: Store Key Assignments Command: Defines multiple keys on the user keyboard by assigning objects to specified keys.

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).

The optional initial list parameter or argument S restores all keys without user assignments to their *standard* key assignments on the user keyboard. This is meaningful only when all standard key assignments had been suppressed (for the user keyboard) by the command S DELKEYS (see DELKEYS).

If the argument *obj* is the name SKEY, the specified key is restored to its *standard key* assignment.

Access: (CAT) STOKEYS

Input/Output:

Level 1/Argument 1		Level 1/Item 1
{ obj_1, $x_{key\ 1}$, ... obj_n, $x_{key\ n}$ }	→	
{ S, obj_1, $x_{key\ 1}$, ... obj_n, $x_{key\ n}$ }	→	
'S'	→	

See also: ASN, DELKEYS, RCLKEYS

STOΣ

Type: Command

Description: Store Sigma Command: Stores *obj* in the reserved variable Σ*DAT*.

STOΣ accepts any object anFd stores it in Σ*DAT*. However, if the object is not a matrix or the name of a variable containing a matrix,, an Invalid ΣDATA error occurs upon subsequent execution of a statistics command.

Access: CAT STOΣ

Input/Output:

Level 1/Argument 1		Level 1/Item 1
obj	→	

See also: CLΣ, RCLΣ, Σ+, Σ–

STR→

Type: Command

Description: Evaluate String Command: Evaluates the text of a string as if the text were entered from the command line.

OBJ→ also includes this function.

Access: CAT STRING→

Input/Output:

Level 1/Argument 1		Level 1/Item 1
"*obj*"	→	*evaluated-object*

See also: ARRY→, DTAG, EQ→, LIST→, OBJ→, →STR

→STR

Type: Command

Description: Object to String Command: Converts any object to string form.

The full-precision internal form of a number is not necessarily represented in the result string. To ensure that →STR preserves the full precision of a number, select Standard number display format or a wordsize of 64 bits (or both) before executing →STR.

The result string includes the entire object, even if the displayed form of the object is too large to fit in the display.

If the argument object is normally displayed in two or more lines, the result string will contain newline characters (character 10) at the end of each line. The newlines are displayed as the character ■.

If the argument object is already a string, →STR returns the string.

Access: CAT →STRING

Input/Output:

Level 1/Argument 1		Level 1/Item 1
obj	→	"*obj*"

See also: →ARRY, →LIST, STR→, →TAG, →UNIT

STREAM

Type: Command

Description: Stream Execution Command: Moves the first two elements from the list onto the stack, and executes *obj*. Then moves the next element (if any) onto the stack, and executes *obj* again using the previous result and the new element. Repeats this until the list is exhausted, and returns the final result.

STREAM is nominally designed for *obj* to be a program or command that requires two arguments and returns one result.

Access: ↰ PRG LIST PROCEDURES STREAM

Input/Output:

Level 2/Argument 1	Level 1/Argument 2		Level 1/Item 1
{ *list* }	*obj*	→	*result*

See also: DOSUBS

STWS

Type: Command

Description: Set Wordsize Command: Sets the current binary integer wordsize to *n* bits, where *n* is a value from 1 through 64 (the default is 64).

Values of *n* less than 1 or greater than 64 are interpreted as 1 or 64, respectively.

If the wordsize is smaller than an integer entered on the command line, then the *most* significant bits are not displayed upon entry. The truncated bits are still present internally (unless they exceed 64), but they are not used for calculations and they are lost when a command uses this binary integer as an argument.

Results that exceed the given wordsize are also truncated to the wordsize.

Access: ↰ MTH BASE STWS

↱ BASE STWS

Input/Output:

Level 1/Argument 1		Level 1/Item 1
n	→	
$\#n$	→	

See also: BIN, DEC, HEX, OCT, RCWS

SUB

Type: Command Operation

Description: Subset Command: Returns the portion of a string or list defined by specified positions, or returns the rectangular portion of a graphics object or *PICT* defined by two corner pixel coordinates.

If $n_{\text{end position}}$ is less than $n_{\text{start position}}$, SUB returns an empty string or list. Values of n less than 1 are treated as 1; values of n exceeding the length of the string or list are treated as that length.

For graphics objects, a user-unit coordinate less than the minimum user-unit coordinate of the graphics object is treated as that minimum. A pixel or user-unit coordinate greater than the maximum pixel or user-unit coordinate of the graphics object is treated as that maximum.

Access: ↰ PRG LIST SUB

Input/Output:

Level 3/Argument 1	Level 2/Argument 2	Level 1/Argument 3		Level 1/Item 1
$[[\ matrix\ ]]_1$	$n_{\text{startposition}}$	$n_{\text{endposition}}$	→	$[[\ matrix\ ]]_2$
$[[\ matrix\ ]]_1$	$\{n_{\text{row}}, n_{\text{column}}\}$	$n_{\text{endposition}}$	→	$[[\ matrix\ ]]_2$
$[[\ matrix\ ]]_1$	$n_{\text{startposition}}$	$\{n_{\text{row}}, n_{\text{column}}\}$	→	$[[\ matrix\ ]]_2$
$[[\ matrix\ ]]_1$	$\{n_{\text{row}}, n_{\text{column}}\}$	$\{n_{\text{row}}, n_{\text{column}}\}$	→	$[[\ matrix\ ]]_2$
$\text{“}string_{\text{target}}\text{”}$	$n_{\text{startposition}}$	$n_{\text{endposition}}$	→	$\text{“}string_{\text{result}}\text{”}$
$\{\ list_{\text{target}}\ \}$	$n_{\text{startposition}}$	$n_{\text{endposition}}$	→	$\{\ list_{\text{result}}\ \}$
$grob_{\text{target}}$	$\{\#n_1, \#m_1\}$	$\{\#n_2\ \#m_2\}$	→	$grob_{\text{result}}$
$grob_{\text{target}}$	(x_1, y_1)	(x_2, y_2)	→	$grob_{\text{result}}$
$PICT$	$\{\#n_1, \#m_1\}$	$\{\#n_2\ \#m_2\}$	→	$grob_{\text{result}}$
$PICT$	(x_1, y_1)	(x_2, y_2)	→	$grob_{\text{result}}$

See also: CHR, GOR, GXOR, NUM, POS, REPL, SIZE

SVD

Type: Command

Description: Singular Value Decomposition Command: Returns the singular value decomposition of an $m \times n$ matrix.

SVD decomposes A into 2 matrices and a vector. U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix, and S is a real vector, such that $A = U \times diag(S) \times V$. S has length MIN(m,n) and contains the singular values of A in nonincreasing order. The matrix $diag(S)$ is an $m \times n$ diagonal matrix containing the singular values S.

The computed results should minimize (within computational precision):

$$\frac{|A - U \cdot \mathrm{diag}(S) \cdot V|}{\min(m, n) \cdot |A|}$$

where diag(S) denotes the $m \times n$ diagonal matrix containing the singular values S.

Access: ↰ MATRICES FACTORIZATION SVD

↰ MTH MATRIX FACTORS SVD

Input/Output:

Level 1/Argument 1		Level 3/Item 1	Level 2/Item 2	Level 1/Item 3
[[*matrix*]]$_A$	→	[[*matrix*]]$_U$	[[*matrix*]]$_V$	[*vector*]$_S$

See also: DIAG→, MIN, SVL

SVL

Type: Command

Description: Singular Values Command: Returns the singular values of an $m \times n$ matrix.

SLV returns a real vector that contains the singular values of an $m \times n$ matrix in non-increasing order. The vector has length MIN(m,n).

Access: ↰ MATRICES FACTORIZATION SVL

↰ MTH MATRIX FACTORS SVL

Input/Output:

Level 1/Argument 1		Level 1/Item 1
[[*matrix*]]	→	[*vector*]

See also: MIN, SVD

SWAP

Type: RPL Command

Description: Swap Objects Command: Swaps the position of the two inputs.

Access: ↰ PRG STACK SWAP

Input/Output:

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

See also: DUP, DUPN, DUP2, OVER, PICK, ROLL, ROLLD, ROT

SYSEVAL

Type: Command

Description: Evaluate System Object Command: Evaluates unnamed operating system objects specified by their memory addresses.

Using SYSEVAL with random addresses can corrupt memory.

Access: CAT SYSEVAL

Input/Output:

Level 1/Argument 1		Level 1/Item 1
$\#n_{\text{address}}$	→	

See also: EVAL, LIBEVAL, FLASHEVAL
