The following quick reference guide is a condensation of the command reference in the *TI-89/TI-92 Plus Guidebook*. It consists of the following sections:

- Summary of commands
- Reserved system variable names
- EOS (Equation Operating System) Hierarchy

I have included this information in the tip list to save you the trouble of referring to the guidebook while using the tip list. It may also be useful in its own right, if printed out at reduced size to be carried with your calculator.

Note that the codes for `setMode()`, `getMode()`, `setGraph()` and `setTable()` are shown in tables in the respective function definitions. These tables include the more recent numeric string codes.
Quick reference guide
to functions and commands

The function or command name is shown in bold text as the first line of the description. The lines immediately following show the arguments, if any. Arguments are shown in italic text. Optional arguments are shown in square brackets [ ]. expr is an expression, var is a variable. If a function can accept more than one type of argument, they may be shown on one line separated by commas, for example,

(expr), (list), (matrix)

If a function may optionally take no arguments, that is indicated as (). Program and function structures may be shown with statements separated with the colon :, which is the symbol actually used for that purpose in programs. Functions and commands indicated with symbols are shown at the end of the reference guide.

abs()
(expr), (list), (matrix)
Return absolute value of real argument or modulus of complex argument.

and
expr1 and expr2
list1 and list2
matrix 1 and matrix 2
Return true or false or a simplified form.

integer1 and integer2
Bit-by-bit 32-bit integer compare. Arguments larger than 32-bit signed values are reduced with symmetric modulo operation. Mode must be Auto or Exact.

AndPic picVar [, row, column]
Logical-AND of graph screen and picvar at (row, column); default is (0,0)

angle()
(expr), (list), (matrix)
Return angle of arguments interpreted as complex numbers. Undefined variables are treated as real variables.

ans()
(), (integer)
Return answer from home screen. integer can be 1 to 99; not an expression; default is zero.

approx()
(expr), (list), (matrix)
Evaluate argument as decimal value if possible.

Archive var1[,var2] [,var3] ...
Move variables to flash ROM.

arcLen()
(expr.var.start,end), (list.var.start,end)
Return arc length of expression (or each element of list) with respect to var from start to end.

augment()
(list1.list2)
Return list which is list2 appended to list1

(matrix1,matrix2)
Append matrix2 appended to matrix1 as columns.

(matrix1;matrix2)
Append matrix2 to matrix1 as rows.

avgRC(expr.var [,h])
Return forward-difference quotient (average rate of change) of expr with respect to var. h is the step value; if omitted, h=0.001

→Bin
integer1→Bin
Convert integer1 to binary. Non-decimal integers must be preceded with 0b or 0h. Arguments larger than 32-bit signed values are reduced with symmetric modulo operation.

BldData [datavar]
Create the data variable datavar based on the current graph settings. If datavar is omitted, the system variable sysdata is used. The variable is a table of function values evaluated at each plot point.

cFactor()
(expr[,var])
Return the nearest integer that is greater than or equal to expr or each element of list or matrix. Argument may be real or complex.
(list[,var]), (matrix[,var])
Return complex factors of argument over a
common denominator. If var is omitted,
argument is factored with respect to all
variables. If var is used, cfactor() tries to factor
the argument toward factors which are linear in
var.

char(integer)
Return string of character indicated by integer.
Integer must be in the range 0 to 255.

Circle x, y,r [,drawmode]
Draw a circle on the Graph with radius r and
center (x, y). All arguments are in window
coordinate units.
drawmode = 1: draw circle (default)
drawmode = 0: turn off the circle
drawmode = -1: invert circle pixels

ClrDraw
Clear the Graph screen; reset Smart Graph

ClrErr
Clear error status and internal error context
variables. Sets errnum to zero.

ClrGraph
Clears graphed functions or expressions.

ClrHome
Clears the home screen, and sets arbitrary
constant suffixes to 1.

ClrIO
Clears the program I/O screen.

ClrTable
Clears table settings which apply to the ASK
setting in the Table Dialog setup box.

colDim(matrix)
Return number of rows of matrix.

colNorm(matrix)
Return maximum of sums of absolute values of
matrix column elements.

comDenom()
(expr [,var])
(list [,var])
(matrix [,var])
Return reduced ratio of expanded numerator
over expanded denominator, with respect to var
if used. Using var can save time, memory and
screen space, and result in expressions on
which further operations are less likely to result
in Memory errors.

conj()
(expr), (list), (matrix)
Return the complex conjugate of the argument.
Undefined variables are treated as real.

CopyVar
var1, var2
Copy the contents of var1 to var2. Unlike the
store operation (→), CopyVar does not simplify
the source variable. CopyVar must be used with
non-algebraic variable types such as Pic and
GDB.

cos()
(expr), (list)
Return the cosine of the argument. You may
override the current angle mode with ° or ′.

(matrix)
Return the matrix cosine of square
diagonalizable matrix, which is not the cosine of
each element. Symbolic elements must have
assigned values. The matrix cosine is calculated
with floating-point arithmetic (Approx mode).

cos⁻¹()
(expr), (list)
Return the angle whose cosine is expression.

(matrix)
Return the matrix inverse cosine of square
diagonalizable matrix, which is not the inverse
cosine of each element. Results are found with
floating-point arithmetic.

cosh()
(expr), (list)
Return the hyperbolic cosine of the argument.

(matrix)
Return the hyperbolic cosine of square
diagonalizable matrix, which is not cosh() of
each element. Results are found with
floating-point arithmetic.

cosh⁻¹()
(expr), (list)
Return the inverse hyperbolic cosine of the
argument.
Return the inverse hyperbolic cosine of square diagonalizable matrix, which is not $\cosh^{-1}()$ of each element. Results are found with floating-point arithmetic.

crossP()
(list1,list2), (matrix1,matrix2)
Return the cross product of the arguments. Lists must have equal dimensions of 2 or 3. Matrices must both be either row or column vectors of equal dimensions 2 or 3.

cSolve(equation, var)
Return real and complex solutions for var. Complex results are possible in Real mode. Fractional powers with odd denominators use the principle branch, not the real branch. Specify complex variables with the underscore '_' suffix.

(cSolve(equation1 and equation2 [and ...]),
(var1,var2[...]))
Return real and complex solutions for var to system of equations equation1, equation2, ... var may be a variable or a variable with a solution guess such as $x=1+3i$. Not all equation variables need be in the solution variable list. A close complex solution guess may be needed to get a solution.

CubicReg
xlist,ylist,[freqList],[catlLst],[catincList]
Calculate cubic polynomial regression and update all statistics variables. First four arguments must be variable names or c1-c99. Last argument need not be a variable name and cannot be c1-c99. All lists except catincList must have equal dimension. The regression equation is $y = a*x^3 + b*x^2 + c*x + d$.

cumSum()
(list), (matrix)
Return list of cumulative sum of elements or list, or return matrix of cumulative sums of columns from top to bottom.

CustmOff
Remove a custom toolbar.

CustmOn
Activate a custom toolbar as set up by Custom ... EndCustom block.

Custom
Set up a custom toolbar. Similar to ToolBar except that Title and Item statements cannot have labels.

Cycle
Transfer program control immediately to the next iteration of the current For, While or Loop loop.

CyclePic
picString, n [], [wait], [cycles], [direction]
Display each of $n$ picture variables specified by picString for wait seconds. cycles specifies the number of display cycles. direction is set to 1 (default) for a circular cycle or to -1 for a forward-backward cycle. To display three variables pic1, pic2 and pic3, picString is "pic" and $n$ is 3.

-Cylind
vector •Cylind
Display row or column vector in cylindrical form $[r,\theta, z]$. vector must be a row or column vector with three elements.

cZeros(expr, var)
Equivalent to exp(list(cSolve(expr=0, var), var)). Return list of real and complex solutions for var which make expr=0. Use the underscore '_' suffix to specify complex variables. var may be a variable name or a solution guess such as $x = 1 + 3i$.

d(expr, var [, order])
(list, var [, order])
(matrix, var [, order])
Return derivative of order of expr, list or matrix argument with respect to var. order must be an
integer. If order is less than zero, the anti-derivative is returned. d() does not fully simplify the expression and var before finding the derivative.

\[ \text{DD} \]
number\text{DD}  
list\text{DD}  
matrix\text{DD}  
Return decimal equivalent of the angle argument. The current Angle Mode sets the argument as degrees or radians. The argument may be radians.

\[ \text{Dec} \]
number\text{Dec}  
Convert number to a decimal number, regardless of the current Base mode.

\[ \text{Define} \]
Define functionName(arg1,arg2,...)=expr  
Create function functionName() with arguments arg1, arg2. fName() evaluates expression with the supplied arguments and returns the result. Do not use arg1 or arg2 as arguments when calling functionName(). This form of Define is equivalent to \( \text{expr} \rightarrow \text{functionName}(\text{arg1},\text{arg2}) \).

Define functionName(arg1,arg2,...)=Func  
EndFunc  
Same as above except that block can include multiple expressions and use Return to return the result.

Define programName(arg1,arg2,...)=Prgm  
EndPrgm  
Same as above except that program programName() is defined. Programs cannot return results.

\[ \text{DelFold} \]
folder1 [,folder2, folder3, ...]  
Delete folders. The Main folder cannot be deleted. An error message is shown if any folder contains variables.

\[ \text{DelVar} \]
var1 [var2, var3, ...]  
Delete variables from memory.

\[ \text{deSolve} \]
(ode12,indVar,depVar)  
Return general solution of 1st- or 2nd order ordinary differential equation ode12 with independent variable indVar and dependent variable depVar. The prime symbol ' indicates the 1st derivative, two prime symbols ' ' indicate the second derivative.

(ode1 and inCond,indVar,depVar)  
Return particular solution of 1st-order differential equation ode1 with initial condition inCond. inCond is in the form \( \text{depVar}(\text{inIndVar}) = \text{inDepVar} \); for example, \( \text{y}(0) = 1 \). inIndVar and inDepVar can be variables with no stored values.

(ode2 and inCond1 and inCond2,indVar,depVar)  
Return particular solution of 2nd-order differential equation ode2 with initial conditions inCond1 and inCond2. inCond1 specifies the value of the independent variable at a point, in the form \( \text{depVar}(\text{inIndVal}) = \text{inDepVal} \). inCond2 specifies the value of the first derivative at a point in the form \( \text{depVar}'(\text{inIndVal}) = \text{in1stDeriv} \).

(ode2 and BndCnd1 and BndCnd2,indVar,depVar)  
Return particular solution of 2nd-order differential equation with boundary conditions BndCnd1 and BndCnd2.

\[ \text{det} \]
(matrix [,tol])  
Return determinant of square matrix. Any element less than tol is considered zero. The default tol is \( 5E-14 \times \text{max}(\text{dim}(\text{matrix})) \times \text{rowNorm}(\text{matrix}) \).

\[ \text{diag} \]
(list), (rowMatrix), (colMatrix)  
Return matrix with main diagonal elements of argument.

(matrix)  
Return row matrix whose elements are the main diagonal elements of square matrix.

\[ \text{Dialog} \]
Dialog : block : EndDlog  
Display a dialog box during program execution. block consists of Text, Request, DropDown and Title commands. Dialog box variables are displayed as default values. If [ENTER] is pressed, the variables are updated and the system variable ok is set to 1. If [ESC] is pressed, the variables are not updated and ok is set to zero.
**dim(list)**
Return dimension of list.

**(matrix)**
Return list of dimensions of *matrix* as (rows,columns)

**(string)**
Return dimension (number of characters) of *string*.

**Disp [expOrString1], [expOrString2], ...**
Display the program I/O screen. If any arguments (expressions or strings) are used, they are displayed on separate screen lines. Arguments are displayed in Pretty Print if Pretty Print mode is On.

**DispG**
Display the Graph Screen.

**DispHome**
Display the Home screen.

**DispTbl**
Display the Table screen. Use the cursor pad to scroll. Press [ENTER] or [ESC] to resume program operation.

**»DMS**
`expr »DMS`, `list »DMS`, `matrix »DMS`
Display the argument as an angle in format DDDDDD°MM'SS.ss". »DMS converts from radians in radian mode.

**dotP()**
`list1.list2`, `(vector1,vector2)`
Return the dot product of two lists or vectors. Vectors must both be row or column vectors.

**DrawFunc expr**
Draw `expr` as a function of `x` on the Graph screen.

**DrawInv expr**
Draw the inverse of `expr` on the Graph screen by plotting `x` values on the `y` axis and vice versa.

**DrawParm expr1, expr2 [,tmin] [,tmax] [,tstep]**
Draw parametric expressions `expr1` and `expr2` with `t` as the independent variable, from `tmin` to `tmax` with `tstep` between each evaluated `t` value. The current Window variables are the defaults for `tmin, tmax` and `tstep`. Using the `t`-arguments does not change the Window settings. The `t`-arguments must be used if the current Graph mode is not Parametric.

**DrawPol expr, [,θmin] [,θmax] [,θstep]**
Draw `expr` as a polar graph with independent variable `θ`. Defaults for `θmin, θmax` and `θstep` are the current Window settings. Specifying `θ`-arguments does not change the Window settings. The `θ`-arguments must be used if the Graph mode is not Polar.

**DrawSlp x1, y1, slope**
Display the Graph screen and draw the line `y = slope*(x-x1) + y1`.

**DropDown title, {item1, item2, ...},var**
Display a drop-down menu with the name `title` in a Dialog box. The drop-down menu choices are 1: `item2`, 2: `item2` and so on. The number of the selected item is stored in `var`. If `var` exists and its value is in the range of items, the referenced item is the default menu selection.

**DrwCtour expr list**
Draw contours on the current 3D graph, in addition to the contours specified by the `ncontour` system variable. `expr` or `list` specifies the z-values at which the contours are drawn. 3D graph mode must be set. `DrwCtour` sets the graph style to CONTOUR LEVELS.

**e (enter exponent)**
`mantissa*10^exponent`
Enter a number in scientific notation as `mantissa * 10^exponent`. To avoid a decimal value result, use `10^integer`, instead.

**e^()**
`expr`
Raise e to the `expr` power. e is the natural logarithm base 2.718..., not the character 'e'. In Radian angle mode, you may enter complex numbers in the polar format `re^iθ`.

**(list)**
Return list of e raised to each element in `list`.

**(matrix)**
Return the matrix exponential of square `matrix`, which is not the same as e raised to the power of each element. See `cos()` for calculation.
details. *matrix* must be diagonalizable. The result always contains floating-point numbers.

\[
eigVc(matrix)
\]
Return matrix of eigenvectors of square *matrix*, whose elements may be real or complex. The eigenvectors are not unique, but are normalized such that if the eigenvector is

\[
V = [x_1, x_2, ..., x_n]
\]

then

\[
\sqrt{x_1^2 + x_2^2 + ... x_n^2} = 1
\]

\[
eigVl(matrix)
\]
Return the eigenvalues of square diagnonalizable *matrix*, whose elements may be real or complex.

**Else** See If
**ElseIf** See If
**EndCustm** See Custom
**EndDlog** See Dialog
**EndFor** See For
**EndFunc** See Func
**EndIf** See If
**EndLoop** See Loop
**EndPrgm** See Prgm
**EndTBar** See ToolBar
**EndTry** See Try
**EndWhile** See While

\[
entry([integer])
\]
Return previous entry-line expression from the history area. If used, *integer* cannot be an expression and must be in the range 1 to 99. See also ans().

\[
exact() \quad (expr [, tol]), (list [, tol]), (matrix [, tol])
\]
Evaluate argument with Exact mode arithmetic regardless of the current Mode setting. *tol*, if used, specifies the conversion tolerance and is zero by default.

\[
Exec \quad string [ , expr1][ , expr2] ...
\]
Execute *string* interpreted as Motorola 68000 assembly language op-codes. *expr1*and *expr2* are optional input arguments. Misinformed use of *Exec* can lock up the calculator and cause data loss.

\[
Exit
\]
Exit the current For, While or Loop block.

\[
exp\cdot list(expr, var)
\]
Convert *expr* to a list of the right-hand sides of equations separated by 'or' in *expr*. Used to extract individual solutions from results of solve(), cSolve(), fMin() and fMax().

\[
expand() \quad (expr), (list), (matrix)
\]
Expand argument with respect to all variables with transformation into a sum and/or difference of simple terms. See also Factor() and expand().

\[
(expr, var), (list, var), (matrix, var)
\]
Expand argument with respect to *var* by collecting similar powers of *var* and sorting the factors with *var* as the main variable. *expand()* also distributes logarithms and fractional powers regardless of *var*. Specifying *var* can save time, memory and result in a smaller expression. If the argument only contains one variable, specifying it as *var* may result in more complete partial factorization. propFrac() is faster but less extreme than expand(). See also comDenom() and tExpand().

\[
expr(string)
\]
Return the evaluated expression in *string*.

\[
ExpReg \quad xlist, ylist[, [freqList], [catlLst], [catincList]]
\]
Calculate exponential polynomial regression and update all statistics variables. First four arguments must be variable names or c1-c99. Last argument need not be a variable name and cannot be c1-c99. All lists except *catincList* must have equal dimension. The regression equation is \(y = a \cdot b^x\).

\[
factor() \quad (expr), (list), (matrix)
\]
Factor argument with respect to all variables over a common denominator. Argument is factored as much as possible toward linear rational factors without introducing new non-real sub-expressions.

\[
factor() \quad (expr, var), (list, var), (matrix, var)
\]
Factor argument with respect to *var*, as much as possible toward real factors linear in *var*, even if irrational constants or subexpressions are
introduced. Also, comDenom() can achieve partial factoring if factor() is too slow or exhausts memory. See also cFactor()

\( \text{gcd}(\text{number1}, \text{number2}), (\text{list1}, \text{list2}), (\text{matrix1}, \text{matrix2}) \)

Return the greatest common denominator of the arguments. The GCD of two fractions is the GCD of the numerators divided by the least common multiple of the denominators. The GCD of fractional floating-point numbers is 1.0 in Auto or Approx mode. If the arguments are lists or matrices, the GCD of each corresponding element is returned.

\( \text{Get var} \)

Retrieve a CBL or CBR value from the link port and store it in var.

\( \text{format}(\text{expr}, \text{format}) \)

Return expression expression as a string formatted with the numeric format string format:

"F[n]" Fixed format; \( n \) is number of digits after the decimal point.

"S[n]" Scientific format; \( n \) is the number of digits after the decimal point.

"E[n]\[c\]" Engineering format; \( n \) is the number of digits after the first significant digit. Mantissa radix point is adjusted so that exponent is a power of three.

\( \text{gcd}(\text{ratNum}) \)

Return factors of rational number \( \text{ratNum} \). Use isPrime(), instead, to more quickly determine if \( \text{ratNum} \) is prime.

\( \text{floor()} \)

Return greatest integer which is less than or equal to the argument. The argument may be real or complex. floor() is identical to int(). See also ceiling().

\( \text{fMax}(\text{expr}, \text{var}) \)

Return Boolean expression specifying possible values of \( \text{var} \) which maximize \( \text{expr} \) with respect to \( \text{var} \), or locate the least upper bound. Use the “with” operator “|” to limit the solution range or add constraints. In Approx mode, fMax() finds one approximate local maximum.

\( \text{fMin}(\text{expr}, \text{var}) \)

Return Boolean expression specifying possible values of \( \text{var} \) which minimize \( \text{expr} \) with respect to \( \text{var} \), or locate the greatest lower bound. Use the “with” operator “|” to limit the solution range or add constraints. In Approx mode, fMin() finds one approximate local minimum.

\( \text{for} \text{var}, \text{low}, \text{high} [,\text{step}] : \text{block} : \text{EndFor} \)

Execute each statement in block iteratively for each value of \( \text{var} \). \( \text{var} \) ranges from \( \text{low} \) to \( \text{high} \) and increments by \( \text{step} \). \( \text{step} \) may be positive or negative and the default is 1. \( \text{var} \) cannot be a system variable.
GetCalc var
Retrieve a value from the link port from another calculator and store it in var.

GetConfig
Return a list of calculator attribute pairs. The first pair element is a string which specifies the attribute, and the second pair element is the attribute. The attribute names are the same for the TI-89 and the TI-92+, but the attributes may differ. The "Cert. Rev. #" attribute pair appears in the list only if a certificate has been installed.


getNum(expr)
Return numerator of expression reduced to a common denominator.

defendom(expr)
Return the reduced common denominator of expression.

defFold()
Return a string which is the current folder name.

defKey()
Return key code of a pressed key as an integer, or return zero if no key is pressed.

defMode(modeString)
Return a string which is the current setting for modeString.

("ALL")
Return a list of string pairs of all Mode settings. The first element of the string pair is the mode name string; the second element is the mode setting string. See SetMode() for possible settings.


defUnits()
Return a list of strings which specify the default units. Constants, temperature, amount of a substance, luminous intensity and acceleration are not included. The list has the form {"system","cat1","unit1","cat2","unit2",...} where system is the unit system: SI, ENG/US or CUSTOM. The cat strings specify the category, and the unit strings specify the corresponding default units.

deflabel label
Transfer program control to label.
Graph
expr [,var] (function graph)
XExpr,yExpr [, var] (parametric graph)
expr [,var] (polar graph)
expr [,xvar] [,yvar] (3D graph)
Graph the expr argument with the current Graph mode. Expressions created with Graph (or Table) are assigned increasing function numbers starting with 1. Modify or delete them with the [F4] Header function in the table display. The currently selected Y= functions are not graphed. The independent variable of the current graph mode is used if a var argument is omitted. Use ClrGraph to clear the functions, or start the Y= editor to enable the system variables.

▶▶▶▶
Hex
integer1
Hex
Convert integer1 to hexadecimal. Non-decimal integers must be preceded with 0b or 0h. Arguments larger than 32-bit signed values are reduced with symmetric modulo operation.

identity(expr)
Return identity matrix with dimension of expr.

If
If BooleanExpr : statement
Execute single statement if BooleanExpr evaluates to TRUE.

If BooleanExpr then : block :EndIf
Execute block if BooleanExpr evaluates to TRUE.

If BooleanExpr then : block1
Else : block2 :EndIf
If BooleanExpr evaluates to TRUE, execute block1 but not block2; otherwise execute block2 but not block1.

If BooleanExpr1 Then : block1
ElseIf BooleanExpr2 Then : block2
...
ElseIf BooleanExprN Then : blockN
EndIf
Execute block1 only if BooleanExpr1 evaluates to TRUE, execute block2 only if BooleanExpr2 evaluates to TRUE, etc.

imag()
(expr), (list), (matrix)
Return the imaginary part of the argument. All undefined variables are treated as real variables.

Input
[promptString,] var
If no arguments, pause program execution, display the Graph screen and update xc and yc (or rc and θc in Polar mode) by positioning the cursor.

If argument var is used, the program pauses, displays promptString on the Program I/O screen and waits for the entry of an expression. The expression is stored in var. If promptString is omitted, "?" is displayed for the prompt.

InputStr
[promptString,] var
Pause program execution, display promptString on the Program I/O screen and wait for the entry of an expression, which is stored as a string in var. "?" is displayed as a prompt if promptString is omitted.

inString(sourceString,targetString[,start])
Return position at which targetString starts in sourceString. The search for targetString begins at character number start, if start is used. start is 1 by default.

int()
(expr), (list), (matrix)
Return the greatest integer that is less than or equal to the argument, which may be real or complex. int() is identical to floor().

intDiv()
(number1,number2), (list1,list2),
(matrix1,matrix2)
Return the signed integer part of the first argument divided by the second argument.

integrate
See $f()$.

iPart()
(number), (list), (matrix)
Return the integer part of the argument, which may be real or complex.

isPrime(number)
Return True or False to indicate if number is prime. Display error message if number has more than about 306 digits and no factors less than or equal to 1021.
Set up a drop-down menu within Custom ... EndCustm, or ToolBar ... EndTBar blocks. In a Custom block, specifies the text that is pasted. In a ToolBar block, specifies the branch label.

Define a label with name labelName in a program. Goto labelName transfers control to the instruction following labelName. labelName must meet same requirements as variable names.

\( \text{lcm}() \)
\( (\text{number1}, \text{number2}) \)
\( (\text{list1}, \text{list2}) \)
\( (\text{matrix1}, \text{matrix2}) \)
Return the least common multiple of the two arguments. The LCM of two fractions is the LCM of the numerators divided by the GCM of the denominators. The LCM of fractional floating point numbers is their product.

\( \text{left}() \)
\( (\text{sourceString}, \text{num}) \)
Return the leftmost num characters from sourceString.

\( (\text{list}, \text{num}) \)
Return the leftmost num elements of list.

\( (\text{comparison}) \)
Return the left-hand side of the equality or inequality of comparison.

\( \text{limit}() \)
\( (\text{expr}, \text{var}, \text{point}, \text{direction}) \)
\( (\text{list}, \text{var}, \text{point}, \text{direction}) \)
\( (\text{matrix}, \text{var}, \text{point}, \text{direction}) \)
Return the limit of the argument expr, list, or matrix with respect to var at point. The limit is taken from the left if direction is negative, from the right if direction is positive, or from both directions otherwise. The default direction is both.

\( \text{Line} \) \( x\text{Start}, y\text{Start}, x\text{End}, y\text{End}, \text{mode} \)
Display Graph screen and draw a line, including the endpoints, from \((x\text{Start}, y\text{Start})\) to \((y\text{Start}, y\text{End})\) according to mode:
\( \text{mode}=0: \) turn off the line
\( \text{mode}=-1: \) invert the line

\( \text{LineHoriz} \) \( y[, \text{mode}] \)
Display Graph screen and draw a horizontal line at window coordinate \(y\), according to mode:
\( \text{mode}=1: \) draw the line (default)
\( \text{mode}=0: \) turn off the line
\( \text{mode}=-1: \) invert the line

\( \text{LineTan} \) \( \text{expr1}, \text{expr2} \)
Display Graph screen and draw line tangent to \(\text{expr1}\) at point \(x = \text{expr2}\). The independent variable for \(\text{expr1}\) is \(x\).

\( \text{LineVert} \) \( x[, \text{mode}] \)
Display Graph screen and draw a vertical line at window coordinate \(x\), according to mode:
\( \text{mode}=1: \) draw the line (default)
\( \text{mode}=0: \) turn off the line
\( \text{mode}=-1: \) invert the line

\( \text{LinReg} \) \( \text{xlist}, \text{ylist}[, [\text{freqList}], [\text{catLst}],[\text{catincList}]] \)
Calculate linear regression and update all statistics variables. First four arguments must be variable names or c1-c99. Last argument need not be a variable name and cannot be c1-c99. All lists except \text{catincList} must have equal dimension. The regression equation is \(y = a \cdot x + b\).

\( \text{list} \cdot \text{mat}(\text{list}, \text{rowElements}) \)
Create a matrix whose elements are filled row-by-row from rowElements specifies the number of elements in each matrix row, if used. The default is one matrix row. If list does not fill the matrix, zeros are added.

\( \text{Δlist}(\text{list}) \)
Return a list which is the difference between successive elements of list. Each element of list is subtracted from the next element, so the returned list is always one element shorter than list.

\( \text{ln}() \)
\( (\text{expr}), (\text{list}) \)
Return the natural logarithm of the argument.

\( \text{ln}() \) \( \text{matrix} \)
Return the matrix natural logarithm of square diagonalizable \text{matrix}. This is not the same as finding the logarithm of each matrix element.
Floating-point results are used. See cos() for calculation details.

**LnReg**
xlist, ylist, [freqList], [catlLst], [catincList]
Calculate logarithmic regression and update all statistics variables. First four arguments must be variable names or c1-c99. Last argument need not be a variable name and cannot be c1-c99. All lists except catincList must have equal dimension. The regression equation is \( y = a + b \cdot \ln(x) \).

**Local**
var1 [. var2, var3, ...]
Declare the arguments as local variables in a program or function. Local variables exist only during program or function execution. Local variables must be used for loop index variables and for storage in multi-line functions, since global variables are not allowed in functions.

**Lock**
var1 [. var2, var3, ...]
Lock the argument variables. Locked variables cannot be changed or deleted unless Unlock() is used.

**log()**
(expn), (list)
Return the base-10 logarithm of the argument.

(matrix)
Return the matrix base-10 logarithm of square diagonalizable matrix. This is not the same as finding the logarithm of each matrix element. Floating-point arithmetic is used. Refer to cos() for calculation details.

**Logistic**
xlist, ylist, [freqList], [catlLst], [catincList]
Calculate logistic regression and update all statistics variables. First four arguments must be variable names or c1-c99. Last argument need not be a variable name and cannot be c1-c99. All lists except catincList must have equal dimension. The regression equation is \( y = \frac{a}{1 + b \cdot e^{-c \cdot x}} + d \) where e is the natural logarithm base.

**Loop**
Loop : block : EndLoop
Repeatedly execute block until a Goto or Exit instruction is executed.

**LU**
matrix, lMat, uMat, pMat [, tol]
Calculate the Doolittle LU (lower-upper) matrix decomposition or real or complex matrix. matrix must be a matrix name. The results are lMat = lower triangular matrix uMat = upper triangular matrix pMat = permutation matrix such that lMat * uMat = pMat * matrix Any matrix element is set to zero if its value is less than tol and matrix contains no floating-point or symbolic elements. The default value for tol is 5E-14 * max(dim(matrix)) * rowNorm(matrix). Computations are done with floating-point arithmetic in Approx mode.

**mat\-list**
(matrix)
Return a list whose elements are those of matrix, row by row.

**max()**
(expr1, expr2), (list1, list2), (matrix1, matrix2)
Return the maximum of the arguments. List and matrix arguments are compared element by element.

((list))
Return the maximum element of list.

(matrix)
Return a row vector whose elements are the maximum elements of the columns of matrix.

**mean(list[, freqList])**
Return the mean of the elements of list. freqList indicates the number of occurrences of the corresponding list element.

((matrix[, freqMatrix])
Return a row vector whose elements are the means of the matrix columns. freqMatrix indicates the number of occurrences of the corresponding matrix element.

**median()**
(list)
Return the median of the elements of list. All list elements must simplify to numbers.

((matrix)
Return a row vector whose elements are the medians of the columns of matrix. All matrix elements must simplify to numbers.
MedMed xlist, ylist, [freqList], [catlLst], [catincList]
Calculate median-median regression and update all statistics variables. First four arguments must be variable names or c1-c99. Last argument need not be a variable name and cannot be c1-c99. All lists except catincList must have equal dimension. The regression equation is \( y = a \times x + b \).

mid()
For both variations of mid(): if count is omitted or greater than the dimension of the argument, the complete argument is returned. count must be greater than or equal to zero. If count = 0, an empty string or list is returned.

\( (\text{string}, \text{start}[, \text{count}]) \)
Return count characters from string beginning at character position start.

\( (\text{list}, \text{start}[, \text{count}]) \)
Return count elements of string beginning at element start.

min()
\( (\text{expr1}, \text{expr2}), (\text{list1}, \text{list2}), (\text{matrix1}, \text{matrix2}) \)
Return the minimum of the arguments. List and matrix arguments are compared element by element.

\( (\text{list}) \)
Return the minimum element of list.

\( (\text{matrix}) \)
Return a row vector whose elements are the minimum elements of the columns of matrix.

mod()
\( (\text{expr1}, \text{expr2}), (\text{list1}, \text{list2}), (\text{matrix1}, \text{matrix2}) \)
Return the first argument modulo the second argument where
\[
\text{mod}(x, 0) = x \\
\text{mod}(x, y) = x - y \times \text{floor}(x/y)
\]
The result is periodic in the second argument when that argument is non-zero. The result is either zero or has the same sign as the second argument.

MoveVar var, curFolder, newFolder
Move var from curFolder to newFolder. newFolder is created if it does not exist.

mRow(expr, matrix, index)
Return matrix with row index of matrix multiplied by expr.

mRowAdd(expr, matrix, index1, index2)
Return matrix with row index2 replaced with expr * row index1 + row index2

nCr(expr1, expr2)
Return number of combinations of expr1 items taken expr2 at a time, for \( expr1 \geq expr2 \geq 0 \), and integer expr1 and expr2. This is the binomial coefficient.

\( (\text{expr}, n) \)
If n is zero or a negative integer, return 0. If n is a positive integer or non-integer, return
\[
\frac{\text{expr}!}{n!(\text{expr} - n)!}
\]

\( (\text{list1}, \text{list2}) \)
\( (\text{matrix1}, \text{matrix2}) \)
Return list of nCr() of corresponding element pairs of each argument. Both arguments must have the same dimension.

nDeriv()
\( (\text{expr}, \text{var}[, \text{h}]) \)
Return an estimate of the numerical derivative of expr with respect to var using the central difference quotient formula. h is the step size and defaults to 0.001.

\( (\text{expr}, \text{var}, \text{list}) \)
Return list of derivative estimates for each step size in list.

\( (\text{list}, \text{var}[, \text{h}]) \)
\( (\text{matrix}, \text{var}[, \text{h}]) \)
Return list or matrix of derivative estimates for each element of list or matrix.

NewData var list1, [list2] [,...] list3 ...
Create data variable var whose columns are the lists in order. The lists can be lists, expressions or list variable names. Make the new variable current in the Data/Matix editor.

var, matrix
Create data variable var based on matrix.
sysdata, matrix
Load matrix in the sysdata system data variable.

NewFold folderName
Create a folder with the name folderName, and make that folder current.

newList(expr)
Return a list of zeros with dimension expr.

newMat(exprRow, exprCol)
Return a matrix of zeros with exprRow rows and exprCol columns.

NewPic matrix, var [, maxRow] [, maxCol]
Create a pic variable var based on matrix. matrix has two columns, and each row is the coordinates of a pixel which is turned 'on' in the picture. var is replaced if it exists. The default size of var is the minimum boundary area required by the matrix coordinates. maxRow and maxCol specify optional maximum boundary limits for var.

NewPlot n, type, xlist[, ylist[, frqList[, catList[, incList[, mark][, bucketSize]]]]]
Create a new data plot definition for plot number n, where n is 1 to 9. xlist and ylist are the data to plot. frqList is the frequency list, catList is the category list and incList is the category include list. type specifies the plot type:
1 = scatter plot
2 = xyline plot
3 = box plot
4 = histogram
5 = modified box plot

mark specifies the plot symbol:
1 = box
2 = cross
3 = plus
4 = square
5 = dot

bucketSize must be > 0 and specifies the histogram bucket width and varies with xmin and xmax. bucketSize defaults to 1. incList need not be a variable name and cannot be c1 - c99. The other list arguments must be variable names or c1 - c99.

NewProb
Clear all unlocked and unarchived single-character variable names in the current folder.

Turn all function and stat plots in the current graph mode. Perform ClrDraw, ClrErr, ClrGraph, S1rHome, ClrIO and ClrTable.

nInt(expr, var, lower, upper)
Return approximate integral of expr with respect to var with integration limits lower and upper. var must be the only variable in expr. lower and upper must be constants, +∞ or -∞. The adaptive algorithm uses weighted sample values of the integrand in the interval lower<var<upper. nInt() attempts an accuracy of six significant digits and displays the warning message "Questionable accuracy" if the accuracy may be less. Nested nInt() calls can be used for multiple integration, and the integration limits can depend on integration variables outside them.

norm(matrix)
Return the Forbenius norm of matrix.

not not BooleanExp
not integer
not list
not matrix
For Boolean arguments, return true, false or a simplified Boolean expression. For integer arguments, return the 1's complement of integer. Results are displayed according to the Base mode. Use the prefixes 0b or 0h for binary and hexadecimal integers. If the integer argument is larger than the maximum signed 32-bit binary number, a symmetric modulo operation brings the argument into range. Auto or Exact mode must be used with integer arguments.

nPr(expr1, expr2)
For integer expr1 and expr2, and expr1 ≥ expr2 ≥ 0, return the number of permutations of expr1 items taken expr2 at a time.

(m, r)
If r = 0, return 1. Otherwise return m!/(m-r)!

nSolve(eqn, var)
Solve for eqn for one approximate solution of var. var may be a variable name in eqn, or a variable name with a solution guess in the form var = guess. Use the "|" operator to constrain the solution range. nSolve() will return the string "no solution found" if it cannot find a plausible solution.
OneVar \texttt{xList}, [\texttt{freqList}], [\texttt{catList}], [\texttt{incList}]
Calculate one-variable statistics and update system statistics variables. All lists must have equal dimension except \texttt{incList}. The first three arguments must be variable names of c1 - c99. \texttt{incList} need not be a variable name and cannot be c1 - c99.

or
\texttt{exp1} or \texttt{exp2}
\texttt{list1} or \texttt{list2}
\texttt{matrix 1} or \texttt{matrix 2}
Return \texttt{true} or \texttt{false} or a simplified form.

\texttt{integer1} and \texttt{integer2}
Bit-by-bit 32-bit integer logical or. Arguments larger than 32-bit signed values are reduced with symmetric modulo operation. Mode must be Auto or Exact.

\texttt{ord()}
\texttt{(string)}, \texttt{(list)}
Return numeric code of the first character in \texttt{string}. Return list of numeric codes of first characters of strings in \texttt{list}.

Output \texttt{row, column, exprOrString}
Display expression or string \texttt{exprOrString} at text coordinates \texttt{row}, \texttt{column}. \texttt{exprOrString} can include conversion operations such as \texttt{▶DD}. \texttt{exprOrString} is pretty-printed if Pretty Print is on.

\texttt{P•Rx()}
\texttt{(rExpr, iExpr)}, \texttt{(rList, iList)}, \texttt{(rMatrix, iMatrix)}
Return the x-coordinate of the \texttt{(r,i)} pair \text{i} is interpreted in degrees or radians according to the current angle mode. Use ° or \texttt{r} to over-ride the current mode setting.

\texttt{P•Ry()}
\texttt{(rExpr, iExpr)}, \texttt{(rList, iList)}, \texttt{(rMatrix, iMatrix)}
Return the y-coordinate of the \texttt{(r,i)} pair \text{i} is interpreted in degrees or radians according to the current angle mode. Use ° or \texttt{r} to over-ride the current mode setting.

\texttt{part(expr, n)}
Extract subexpressions of expression \texttt{expr}. \text{n} is a non-negative integer.

Simplify \texttt{expr} and return the number of top-level arguments or operands. Return 0 if \texttt{expr} is a number, variable or symbolic constant.

\( (\texttt{expr, 0}) \)
Simplify \texttt{expr} and return a string which contains the top-level function name or operator. Return string(\texttt{expr}) if \texttt{expr} is a number, variable or symbolic constant.

\( (\texttt{expr, n}) \)
Simplify \texttt{expr} and return the \text{n}th argument or operand, where \text{n} is greater than zero and less than or equal to the number of operands returned by part(\texttt{expr}). Otherwise return an error.

\texttt{PassErr}
Pass an error to the next level. Typically used in an Else clause (in place of ClrErr), when error handling is not yet determined.

\texttt{Pause [expr]}
Suspend program execution until [ENTER] is pressed. If included, \texttt{expr} is displayed on the Program I/O screen. If \texttt{expr} does not fit on the screen, the cursor pad scrolls the display. \texttt{expr} can include conversion operation suffixes such as \texttt{▶DD}.

\texttt{PlotsOff[1] [2] [3] ... [9]}
Turn all plots off with no arguments, or turn the specified plots off. Only affects the current graph in 2-graph mode.

\texttt{PlotsOn[1] [2] [3] ... [9]}
Turn all plots on with no arguments, or turn the specified plots on. Only affects the current graph in 2-graph mode.

\texttt{P•Polar}
\texttt{vector •Polar}
Display vector in polar form \texttt{[r∠θ]}. \texttt{vector} must be a row or column vector of dimension 2. Can only be used at the end of an entry line, and does not update \texttt{ans}.

\texttt{complexValue •Polar}
Display \texttt{complexValue} in polar form: \texttt{[r∠θ]} in Degree mode, or \texttt{re} in Radian mode. \texttt{complexValue} can have any complex form, but \texttt{re} causes an error in Degree mode.
polyEval(list, expr), (list, list2)
Interpret list as the coefficients of a polynomial in descending degree, and return the polynomial evaluated for expr or each element of list2.

PopUp itemList, var
Display a pop-up menu with the character strings in itemList, wait for the press of a number key, and store the number in var. If var exists and contains a valid item number, that item is the default choice. itemList must have at least one character string.

PowerReg xlist, ylist, [freqList], [catlList], [catincList]
Calculate power regression and update all statistics variables. First four arguments must be variable names or c1-c99. Last argument need not be a variable name and cannot be c1-c99. All lists except catincList must have equal dimension. The regression equation is \( y = ax^b \).

Prgm Prgm : block : EndPrgm
Required instruction that identifies the first line of a TI-Basic program.

product(list[,start[,end]])
Return the product of the elements of list, from start to end.

(matrix[,start[,end]])
Return row vector whose elements are the product of the column elements of matrix. start and end specify a row range.

Prompt var1[,var2][,var3]...
Display one prompt for each argument on the Program I/O screen. The prompt is the variable name suffixed with “?” . The expression entered for each prompt is stored in the variable.

propFrac(rationalNumber)
Return rationalNumber as an integer and fraction with the same sign, where the fraction denominator is greater than the numerator.

(prop(rationalExpr[,var])
Return rationalExpr as a sum of proper ratios and polynomial in var. The degree of var in each ratio denominator is greater than the degree in the numerator. Similar powers of var are collected, and the terms and powers are sorted with respect to var.

(rationalExpr)
Return a proper fraction expansion with respect to the most main variable. The polynomial coefficients are then made proper with respect to their most main variable, and so on.

PtChg x,y
xList, yList
Display Graph screen and reverse pixel nearest to window coordinates (x,y).

PtOff x,y
xList, yList
Display Graph screen and turn off pixel nearest window coordinates (x,y).

PtOn x,y
xList, yList
Display Graph screen and turn on pixel nearest window coordinate (x,y).

PtTest()
(x,y), (xList, yList)
Return True if the pixel nearest window coordinates (x,y) is on, otherwise return False.

PtText string, x,y
Display Graph screen and place string with the upper-left corner of the first character at the pixel nearest window coordinates (x,y)

PxlChg row, col
rowList, colList
Display Graph screen and reverse the pixel at pixel coordinates (row, col) or (rowList, colList)

PxlCrcl row, col, r[,drawMode]
Display Graph screen and draw circle with radius r pixels and center at pixel coordinates (row, col). If drawMode = 1, draw the circle (default). If drawMode = 0, turn off the circle. If drawMode = -1, invert pixels along the circle .

PxlHorz row, col, r[,drawMode]
Display Graph Screen and draw horizontal line at pixel coordinate row. If drawMode = 1, draw the line (default). If drawMode = 0, turn the line pixels off. If drawMode = -1, invert the line pixels.
PxlLine
\text{rowStart}, \text{colStart}, \text{rowEnd}, \text{colEnd}[\text{drawMode}]
Display Graph screen and draw line from pixel coordinates \((\text{rowStart}, \text{colStart})\) to \((\text{rowEnd}, \text{colEnd})\), including both endpoints.
If \text{drawMode} = 1, draw the line (default).
If \text{drawMode} = 0, turn the line pixels off.
If \text{drawMode} = -1, invert the line pixels.

PxlOff
\text{row}, \text{col}
Display the Graph screen and turn off the pixel at pixel coordinates \((\text{row}, \text{col})\).

PxlOn
\text{row}, \text{col}
Display Graph screen and turn on the pixel at pixel coordinates \((\text{row}, \text{col})\).

PxlTest() \((\text{row}, \text{col})\), \((\text{rowList}, \text{colList})\)
Return True if the pixel at pixel coordinates \((\text{row}, \text{col})\) is on, otherwise return False.

PxlVert \text{col}[\text{drawMode}]
Display Graph Screen and draw vertical line at pixel coordinate \text{col}.
If \text{drawMode} = 1, draw the line (default).
If \text{drawMode} = 0, turn the line pixels off.
If \text{drawMode} = -1, invert the line pixels.

QR \text{matrix}, \text{qMatName}, \text{rMatName}, [\text{tol}]
Calculate Householder QR factorization of real or complex matrix. The Q and R result matrices are stored to the \text{matName} variables: Q is unitary, R is upper triangular. If the matrix has floating-point elements and no symbolic variables, then any matrix element less than tol is set to zero. The default value for tol is 5E-14 * \max(\text{dim(matrix)}) * \text{rowNorm}(matrix). Computations are done with floating-point arithmetic if the mode is Approximate.

QuadReg \text{xlist}, \text{ylist}, [\text{freqList}, \text{catList}], [\text{catincList}]
Calculate quadratic polynomial regression and update all statistics variables. First four arguments must be variable names or c1-c99. Last argument need not be a variable name and cannot be c1-c99. All lists except \text{catincList} must have equal dimension. The regression equation is \(y = ax^2 + bx + c\).

QuartReg \text{xlist}, \text{ylist}, [\text{freqList}, \text{catList}], [\text{catincList}]
Calculate quartic polynomial regression and update all statistics variables. First four arguments must be variable names or c1-c99. Last argument need not be a variable name and cannot be c1-c99. All lists except \text{catincList} must have equal dimension. The regression equation is \(y = ax^4 + bx^3 + cx^2 + dx + e\).

R-Pθ() \((\text{xExpr}, \text{yExpr}), (\text{xList}, \text{yList})\)
\((\text{xMatrix}, \text{yMatrix})\)
Return the θ-coordinate of the \((x, y)\) arguments, as either a radian or degree angle, depending on the current Angle mode.

R-Pr() \((\text{xExpr}, \text{yExpr}), (\text{xList}, \text{yList}), (\text{xMatrix}, \text{yMatrix})\)
Return the r-coordinate of the \((x, y)\) arguments.

\text{rand}([\text{n}])
n is a non-zero integer. With no argument, return the next random number between 0 and 1 in the sequence. When \text{n}>0, return random integer in the interval \([1, n]\). When \text{n}<0, return random integer in the interval \([-n, -1]\).

\text{randMat}([\text{numRows}, \text{numColumns}])
Return matrix with dimensions \((\text{numRows}, \text{numColumns})\) whose elements are random integers between -9 and 9.

\text{randNorm}([\text{mean}, \text{sdev}])
Return floating-point number from the standard distribution with \text{mean} and standard deviation \text{sdev}.

\text{randPoly}([\text{var}, \text{order}])
Return a polynomial of \text{order} in \text{var} whose coefficients are random integers from -9 to 9, where the leading coefficient is not zero.

\text{RandSeed} \text{n}
Reseed the random number generator. If \text{n} = 0, reseed with the factory defaults. Otherwise, \text{n} is
used to generate the two system variables
seed1 and seed2.

**RclGDB GDBvar**
Restore all Graph database settings from the variable GDBvar. See StoGDB for the settings.

**RclPic picVar,[row,column]**
Display Graph screen and add the picture picVar at the upper-left corner coordinates of (row,column), by logically OR-ing picVar with the Graph screen. The default coordinates are (0,0).

**real()**
(expr), (list), (matrix)
Return the real part of expr, or the real parts of the elements of list or matrix. Undefined variables are treated as real. See also **imag()**.

**rect()**
(vector)
Display vector in rectangular form [x,y,z].

**complexValue • Rect**
Display complexValue in rectangular for a + bi. complexValue may have any form, but an r∠θ entry causes an error in Degree mode. For polar entries the form (r∠θ) must be used.

**ref(matrix[,tol])**
Return the row echelon form of matrix. Treat any element as zero if it is less than tol, and the matrix contains floating-point elements and no symbolic elements. The default tol is 5E-14 * max(dim(matrix)) * rowNorm(matrix). Floating-point arithmetic is used in Approx mode.

**remain()**
(expr1,expr2), (list1,list2), (matrix1,matrix2)
Return the remainder of the first argument with respect to the second argument as remain(x,0) = x
remain(x,y) = x-y*iPart(x/y)
Note that remain(-x,y) = -remain(x,y).

**Rename oldVarName,newVarName**
Rename variable oldVarName as newVarName.

**Request** promptString, var
If used inside a Dialog...EndDlog structure, create a user input box in the dialog box, otherwise create a dialog box with the input box. promptString must be less than 21 characters. If var contains a string, it is displayed as the default.

**Return [expression]**
Exit a program or function if no argument used, otherwise return expression from a function.

**right()**
(list[,num])
Return the rightmost num elements of list. Return list if num not used.

(string[,num])
Return the rightmost num characters of string. Return string if num not used.

(comparison)
Return the right side of equation or inequality.

**rotate()**
(integer[,n])
Rotate signed 32-bit integer n bits. Rotate to the left if n is positive, to the right if n is negative. Default n is -1.

(list[,n])
Return list with elements rotated by n elements. Rotate to the left for positive n, to the right for negative n. Default n is -1.

(string[,n])
Return string with characters rotated by n characters. Rotate to the left for positive n, to the right for negative n. Default n is -1.

**round()**
(expr[,d]), (list[,d]), (matrix[,d])
Return argument elements rounded to d digits after the decimal point. d must be an integer in the range 0-12. Default d is 12.

**rowAdd(matrix,r1,r2)**
Return matrix with row r1 replaced by the sum of rows r1 and r2.

**rowDim(matrix)**
Return number of rows of matrix.
rowNorm(matrix)
Return the maximum of the sums of the row elements of matrix. All elements must simplify to numbers.

rowSwap(matrix,r1,r2)
Return matrix with rows r1 and r2 swapped.

RplcPic picVar(.row[, column]
Place picture picVar in the Graph screen with its upper left corner at pixel coordinates row, column. The area of the Graph screen affected by picVar is cleared. row, column default to 0,0.

rref(matrix[, tol])
Return the reduced row echelon format of matrix. Treat any element as zero if it is less than tol, and the matrix contains floating-point elements and no symbolic elements. The default tol is 5E-14 * max(dim(matrix)) * rowNorm(matrix). Floating-point arithmetic is used in Approx mode.

Send [list]
Send list to the link port; used with CBL and CBR.

SendCalc var
Send var to link port to be received by another calculator. Receiving calculator must be on Home screen or execute GetCalc from a program. Use SendChat, instead, to send from a TI-89/92+ to a TI-92.

SendChat var
Alternative to SendCalc which works with either a TI-92 or TI-92+. Will only send variables which are compatible with the TI-92. Will not send archived variables or graph data base, etc.

seq(expr,var,low,high[,step])
Return list whose elements are expr evaluated at var, for each var from low to high incremented by step. The default for step is 1.

setFold(folderName)
Return the name of the current folder as a string, and set the current folder to folderName. The folder folderName must exist.

setGraph(modeNameString,settingString)
Set the Graph mode modeName to setting and return the previous setting as a string. modeNameString and settingString may be descriptive strings, or numeric code strings. Numeric codes are preferred because the descriptive strings depend on the language localization setting. This table shows both the descriptive string (in English) followed by the equivalent numeric code.

<table>
<thead>
<tr>
<th>modeNameString</th>
<th>settingString</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Coordinates&quot;</td>
<td>&quot;Rect&quot;, &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;1&quot;</td>
<td>&quot;Polar&quot;, &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;Off&quot;, &quot;3&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Graph Order&quot;</td>
<td>&quot;Seq&quot;, &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;2&quot;</td>
<td>&quot;Simul&quot; &quot;2&quot; [a]</td>
</tr>
<tr>
<td>&quot;Grid&quot;</td>
<td>&quot;Off&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;3&quot;</td>
<td>&quot;On&quot; &quot;2&quot; [b]</td>
</tr>
<tr>
<td>&quot;Axes&quot;</td>
<td>&quot;Off&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;4&quot;</td>
<td>&quot;On&quot; &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;Labels&quot;</td>
<td>&quot;Off&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;5&quot;</td>
<td>&quot;On&quot; &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;Seq Axes&quot;</td>
<td>&quot;Time&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;7&quot;</td>
<td>&quot;Web&quot; &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;U1-vs-U2&quot; &quot;3&quot; [d]</td>
<td></td>
</tr>
<tr>
<td>&quot;Solution method&quot;</td>
<td>&quot;RK&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;8&quot;</td>
<td>&quot;Euler&quot; &quot;2&quot; [e]</td>
</tr>
<tr>
<td>&quot;Fields&quot;</td>
<td>&quot;SipFid&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;9&quot;</td>
<td>&quot;DirFid&quot; &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;FidOff&quot; &quot;3&quot; [e]</td>
<td></td>
</tr>
<tr>
<td>&quot;DE Axes&quot;</td>
<td>&quot;Time&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;10&quot;</td>
<td>&quot;t-vs-y&quot;, &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;y-vs-t&quot; &quot;3&quot;</td>
<td>&quot;y1-vs-y2&quot; &quot;4&quot;</td>
</tr>
<tr>
<td>&quot;y1-vs-y2&quot; &quot;5&quot;</td>
<td>&quot;y1-vs-y2&quot; &quot;6&quot;[e]</td>
</tr>
<tr>
<td>&quot;Style&quot;</td>
<td>&quot;Wire Frame&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;11&quot;</td>
<td>&quot;Hidden Surface&quot; &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;Contour Levels&quot;</td>
<td>&quot;3&quot;</td>
</tr>
<tr>
<td>&quot;Wire and Contour&quot;</td>
<td>&quot;4&quot;</td>
</tr>
<tr>
<td>&quot;Implicit Plot&quot; &quot;5&quot;</td>
<td>[c]</td>
</tr>
</tbody>
</table>

[a] Not available in Sequence, 3D or Diff Equations graph mode
[b] Not available in 3D graph mode
[c] Applies only to 3D graph mode
[d] Applies only to Sequence graph mode
[e] Applies only to Diff Equations graph mode
setMode()
(modeString, settingString), (list)
Set mode modeString to settingString, and return the current setting. list contains pairs of modeString and settingString pairs. modeString and settingString may be descriptive strings, or numeric code strings. Numeric codes are preferred because the descriptive strings depend on the language localization setting. Use the list argument to set several modes at once. This table shows both the descriptive string (in English) followed by the equivalent numeric code.

<table>
<thead>
<tr>
<th>modeString</th>
<th>settingString</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ALL&quot;</td>
<td>Only applies to getMode(), not setMode()</td>
</tr>
<tr>
<td>&quot;0&quot;</td>
<td>&quot;Function&quot; &quot;1&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Parametric&quot; &quot;2&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Polar&quot; &quot;3&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Sequence&quot; &quot;4&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;3D&quot; &quot;5&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Diff Equations&quot; &quot;6&quot;</td>
</tr>
<tr>
<td>&quot;Graph&quot;</td>
<td>&quot;Fix 0&quot;, ..., &quot;Fix 12&quot;</td>
</tr>
<tr>
<td>&quot;1&quot;</td>
<td>&quot;Fix n&quot; is &quot;n+1&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Float&quot; &quot;14&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Float 1&quot;, ..., &quot;Float 12&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Float n&quot; is &quot;n+14&quot;</td>
</tr>
<tr>
<td>&quot;Display digits&quot;</td>
<td>&quot;Fix 0&quot;, ..., &quot;Fix 12&quot;</td>
</tr>
<tr>
<td>&quot;2&quot;</td>
<td>&quot;Fix n&quot; is &quot;n+1&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Float&quot; &quot;14&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Float 1&quot;, ..., &quot;Float 12&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Float n&quot; is &quot;n+14&quot;</td>
</tr>
<tr>
<td>&quot;Angle&quot;</td>
<td>&quot;Radian&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;3&quot;</td>
<td>&quot;Degree&quot; &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;Exponential Format&quot;</td>
<td>&quot;Normal&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;4&quot;</td>
<td>&quot;Scientific&quot; &quot;2&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Engineering&quot; &quot;3&quot;</td>
</tr>
<tr>
<td>&quot;Complex Format&quot;</td>
<td>&quot;Real&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;5&quot;</td>
<td>&quot;Rectangular&quot; &quot;2&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Polar&quot; &quot;3&quot;</td>
</tr>
<tr>
<td>&quot;Vector Format&quot;</td>
<td>&quot;Rectangular&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;6&quot;</td>
<td>&quot;Cylindrical&quot; &quot;2&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Spherical&quot; &quot;3&quot;</td>
</tr>
<tr>
<td>&quot;Pretty Print&quot;</td>
<td>&quot;Off&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;7&quot;</td>
<td>&quot;On&quot; &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;Split Screen&quot;</td>
<td>&quot;Full&quot; &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;8&quot;</td>
<td>&quot;Top-Bottom&quot; &quot;2&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Left-Right&quot; &quot;3&quot;</td>
</tr>
<tr>
<td>&quot;Split 1 App&quot;</td>
<td>(no number codes)</td>
</tr>
<tr>
<td>&quot;9&quot;</td>
<td>&quot;Home&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Y= Editor&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Window Editor&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Graph&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Table&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Data/Matrix Editor&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Program Editor&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Text Editor&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Numeric Solver&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Flash App&quot;</td>
</tr>
<tr>
<td>&quot;Split 2 App&quot;</td>
<td>Same as &quot;Split 1 App&quot;</td>
</tr>
</tbody>
</table>

setTable(modeNameString, settingString)
Set table parameter modeNameString to settingString, and return the previous setting. modeNameString and settingString may be descriptive strings, or numeric code strings. Numeric codes are preferred because the descriptive strings depend on the language localization setting. This table shows both the descriptive string (in English) followed by the equivalent numeric code.

<table>
<thead>
<tr>
<th>modeNameString</th>
<th>settingString</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Graph&lt;-&gt;Table&quot;</td>
<td>&quot;Off&quot; &quot;1&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;On&quot; &quot;2&quot;</td>
</tr>
<tr>
<td>&quot;Independent&quot;</td>
<td>&quot;Auto&quot; &quot;1&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Ask&quot; &quot;2&quot;</td>
</tr>
</tbody>
</table>

setUnits(list)
Set default units to values specified by list and return the previous defaults. Specify the built-in SI (metric) units with {"SI"}, or the English/US units with {"ENG/US"}.

Specify a custom set of default units with {"CUSTOM","cat1","unit1","cat2","unit2",...} where "cat" specifies a unit category and "unit" specifies the default unit. Any unspecified category uses the previous custom unit.

Specify the previous custom default units with {"CUSTOM"}

Shade
expr1, expr2, xlow, xhigh, pattern, patRes
Display Graph screen, graphs expr1 and expr2, and shade areas where expr1 < expr2. expr1 < expr2.
and \( \text{expr2} \) use \( x \) as the independent variable. \( x_{\text{low}} \) and \( x_{\text{high}} \) specify left and right shading boundaries, \( x_{\text{low}} \) and \( x_{\text{high}} \) are bounded by, and default to \( x_{\text{min}} \) and \( x_{\text{max}} \).

\( \text{pattern} \) sets the shading pattern:
1 = vertical (default)
2 = horizontal
3 = negative-slope 45°
4 = positive-slope 45°

\( \text{patRes} \) specifies the shading pixel spacing resolution:
1 = solid
2 = 1 pixel spacing (default)
3 = 2 pixel spacing
...
10 = 9 pixel spacing

\( \text{shift()} \)
\( (\text{integer}[,n]) \)
Shift bits in binary \( \text{integer} \), \( n \) times. Shift to left if \( n \) is positive; to right if \( n \) is negative. Default \( n \) is -1. In a right shift, the rightmost bit is dropped, and 0 or 1 is inserted at the left to match the leftmost bit. In a left shift, the leftmost bit is dropped and 0 is inserted as the rightmost bit.

\( (\text{list}[,n]) \)
Return \( \text{list} \) shifted left or right by \( n \) elements. Shift to left if \( n \) is positive, to right if \( n \) is negative. Default \( n \) is -1. Elements introduced by the shift are set to the symbol \( \text{undef} \).

\( (\text{string}[,n]) \)
Return \( \text{string} \) shifted left or right by \( n \) characters. Shift to left if \( n \) is positive, to right if \( n \) is negative. Default \( n \) is -1. Elements introduced by the shift are set to a space.

\( \text{ShowStat} \)
Show dialog box with the last computed statistics results, if the results are still valid. Results are cleared (not valid) if the data to compute them has changed.

\( \text{sign()} \)
\( (\text{expr}), (\text{list}), (\text{matrix}) \)
For real and complex arguments, return \( \text{expr}/\text{abs}(\text{expr}) \) when \( \text{expr} \) is not equal to zero. Return 1 if \( \text{expr} \) is positive, or -1 if \( \text{expr} \) is negative. \( \text{sign}(0) \) returns ±1 in the REAL complex mode, otherwise it returns itself.

\( \text{simult}(\text{coefMatrix},\text{constVector}[,.\text{tol}]) \)
Return column vector with solutions to the system of linear equations where \( \text{coefMatrix} \) is a square matrix of the equation coefficients and \( \text{constVector} \) is a column vector with the same number of rows as \( \text{coefMatrix} \). If the matrix has floating point elements and no symbolic variables, then any element is treated as zero if its value is less than \( \text{tol} \). The default for \( \text{tol} \) is 5E-14 * max(dim(\text{coefMatrix})) * rowNorm(\text{coefMatrix}). Computations are done with floating-point arithmetic in Approx mode.

\( (\text{coefMatrix},\text{constMatrix}[,.\text{tol}]) \)
Solve multiple systems of linear equations, where each system has the same equation coefficients in \( \text{coefMatrix} \). Each column of \( \text{constMatrix} \) contains the constants for a different system of equations. Each column of the returned matrix is the solution for the corresponding \( \text{constMatrix} \) column.

\( \text{sin()} \)
\( (\text{expr}), (\text{list}) \)
Return the sine of the argument. The argument is interpreted in degrees or radians according to the current mode setting. Use ° or \( \degree \) to override the mode setting.

\( (\text{matrix}) \)
Return the matrix sine of square \( \text{matrix} \), which is not the sine of each element. Refer to \( \cos() \) for details. \( \text{matrix} \) must be diagonalizable and the result always contains floating-point numbers.

\( \text{sin}^{-1}() \)
\( (\text{expr}), (\text{list}) \)
Return the angle whose sine is the argument. The result is returned as degrees or radians depending on the current Angle mode setting.

\( (\text{matrix}) \)
Return the matrix inverse sine of square \( \text{matrix} \), which is not the same as the inverse sine of each element. Refer to \( \cos() \) for details. \( \text{matrix} \) must be diagonalizable and the result always contains floating-point numbers.

\( \text{sinh()} \)
\( (\text{expr}), (\text{list}) \)
Return the hyperbolic sine of the argument. The argument is interpreted in degrees or radians according to the current mode setting. Use ° or \( \degree \) to override the mode setting.
Return the matrix hyperbolic sine of square matrix, which is not the hyperbolic sine of each element. Refer to cos() for details. matrix must be diagonalizable and the result always contains floating-point numbers.

\( \sinh^{-1}(\text{expn}) \), \((\text{list})\)
Return the angle whose hyperbolic sine is the argument. The result is returned as degrees or radians depending on the current Angle mode setting.

Return the matrix inverse hyperbolic sine of square matrix, which is not the same as the inverse hyperbolic sine of each element. Refer to cos() for details. matrix must be diagonalizable and the result always contains floating-point numbers.

\( \text{SinReg} \)
\( xlist, ylist, [[\text{iterations}}, [\text{period}}[, \text{catList}, \text{incList}]] \)
Calculate the sinusoidal regression and update the system statistics variables. \( xlist \) and \( ylist \) are the x- and y-data points. \( \text{iterations} \) is the the maximum number of solution interations; the range is 1 to 16 and the default is 8. Larger values may result in better accuracy but longer execution time. \( \text{period} \) specifies the estimated period. If not used, the elements of \( xlist \) should be in sequential order and equally spaced. \( xlist \), \( ylist \) and \( \text{catList} \) must be variable names or c1 - c99. \( \text{incList} \) need not be a variable name and cannot be c1 - c99. All lists must have equal dimensions except \( \text{incList} \).

\( \text{solve()} \)
\( (\text{eqn}, \text{var}), (\text{inequality}, \text{var}) \)
Return candidate real solutions (as Boolean expressions) for \( \text{var} \) of equation \( \text{eqn} \) or \( \text{inequality} \). Attempts to return all solutions, but for some arguments there are infinite solutions. In the Auto mode setting, concise exact solutions are attempted, supplemented by approximate solutions. Solutions may exist only in the limit from one or both sides due to default cancellation of the greatest common divisor from ratio numerators and denominators. \( \text{false} \) is returned if no real solutions can be found. \( \text{true} \) is returned if solve() determines that any finite real value is a solution. Solutions may include unique arbitrary integers of the form @n\(_j\), where \( j \) is an integer from 1 to 255. Use the "|" operator to constrain the solution interval or other variables.

In Real mode, fractional powers with odd denominators denote only the real branch. Otherwise, multiple branched expressions (fractional powers, logarithms, inverse trigonometric functions) denote the principle branch. solve() produces solutions only for that one real or principle branch.

Explicit solutions to inequalities are unlikely unless the inequality is linear and only includes \( \text{var} \). In the Exact mode setting, portions which cannot be solved are returned as implicit equations or inequalities.

\( (\text{eqn1} \text{and} \text{eqn2} \text{[and ...]}, \{\text{var1}, \text{var2}[, ...]\}) \)
Return candidate real solutions (as Boolean expressions) to the simultaneous equations. \( \text{var} \) arguments may be variable names, or variable names with a solution guess in the form \( \text{var} = \text{guess} \). If all equations are polynomials and you supply no guesses, solve() uses the lexical Gröbner/Buchberger elimination to attempt to find all solutions. Simultaneous polynomial equations can have extra variables with no values. Solution variables of no interest may be omitted. Solutions may include arbitrary constants of the form @\( k \), where \( k \) is an integer from 1 to 255. Computation time or memory exhaustion may depend on the order of the \( \text{vars} \) in the equations or variables list.

solve() attempts to find all real solutions with Gaussian elimination if you include no guesses, any equation is in non-polynomial in any variable, but all equations are linear in the solution variables.

solve() attempts to find one real solution (with an interative approximate method) if the system is neither polynomial in all its variables nor linear in its solution variables. The number of solution variables must equal the number of equations and all other variables must simplify to numbers. Each solution variable starts at its guess value, or 0.0 if a guess is not used. Guesses may need to be close to the solution for convergence.

\( \text{SortA} \)
\(\text{listName1}, [\text{listName2}], [\text{listName3}],... \)
vectorName1, vectorName2, vectorName3, ...
Sort the elements of the first argument in ascending order. Sort elements of optional arguments so that the element positions match the new positions of the elements in the first argument. All arguments must be names of lists or vectors, and have equal dimensions.

**SortD**

listName1, listName2, listName3, ...
vectorName1, vectorName2, vectorName3, ...
Same as SortA, but sort the elements in descending order.

**Sphere**

vector
Display 3-element row or column vector in spherical form \([\rho, \theta, \phi]\). \(\theta\) is the angle from the x-axis to the projection of the vector in the xy plane, and \(\phi\) is the angle from the z-axis to the vector.

**stdDev(list, freqList)**

Return the sample (not population) standard deviation of the elements of list. freqList elements specify the frequency of the corresponding elements of list. Both lists must have at least two elements, and the same number of elements.

(matrix, freqMatrix)
Return row vector of the sample standard deviations of the column elements of matrix. freqMatrix elements specify the frequency of corresponding matrix elements. matrix must have at least two rows.

**StoGDB** GDBvar
Create a Graph database variable (GDB) with the current settings for:
Graphing mode; Y= functions; Window variables; Graph format settings (1- or 2-Graph setting; split-screen and ratio settings if 2-Graph mode); Angle mode; Real/Complex mode; Initial conditions (Sequence or Diff Equations mode); Table flags; tblStart, Δtbl, tblInput
Use RclGDB GDBvar to restore the graph environment.

**Stop**
Stop program execution.

**StoPic** picVar, pxlRow, pxlCol, width, height
Display the Graph screen and copy a rectangular area of the display to picVar. pxlRow and pxlCol specify the upper left corner of the copied area, the default is 0,0. width and height specify the pixel dimensions of the area and default to the width and height of the current graph screen.

**Store**
See →

**String(expr)**
Simplify expr and return the result as a string.

**Style eqNum, styleString**
Set graph function eqNum to use the graphing property styleString. eqNum must be 1 - 99 and the function must exist. styleString must be one of: "Line", "Dot", "Square", "Thick", "Animate", "Path", "Above" or "Below". Only some styles are valid for particular graph modes:

Function: all styles
Parametric/Polar: line, dot, square, thick, animate, path
Sequence: line, dot, square, thick
3D: none
Diff Equations: line, dot, square, thick, animate, path.

**subMat()**
(matrix, startRow, startCol, endRow, endCol)
Return specified submatrix of matrix. Defaults are startRow = 1, startCol = 1, endRow = last row, endCol = last column

**sum(list, start, end)**
Return sum of list elements from start to end.

(matrix, start, end)
Return row vector whose elements are the sums of the columns of matrix. start and end specify start and end rows.

**switch([n])**
Return the number of the active window, and switch the active window based on n. Window 1 is top or left, window 2 is right or bottom. With no argument, switch the current window and return the previous active window number. n is ignored if a split screen is not displayed. Otherwise, for n = 0: return the current window.
n = 1: activate window 1, return previous active window number.
n = 2: activate window 2, return previous active window number.

\( \text{\textasciitilde} \) (transpose)
\( \text{matrix}' \)
Return complex conjugate transpose of matrix.

\textbf{Table} \( \text{expr}_1, [\text{expr}_2], [\text{var}] \)
Build a table of the argument expression(s) or functions. The current Y= Editor functions are temporarily ignored. Expressions entered with Table or Graph are assigned increasing function numbers beginning with 1. To clear these functions execute ClrGraph or display the Y= Editor. If \textit{var} is omitted, the current graph mode independent variable is used. Table cannot be used with 3D, sequence or differential equations graphing. Valid variations are:
Function graph: Table \( \text{expr}, x \)
Parametric graph: Table \( x\text{Expr}, y\text{Expr}, t \)
Polar graph: Table \( \text{expr}, \theta \)
See also BldData.

\textbf{tan()}
\( (\text{expr}), (\text{list}) \)
Return the tangent of the argument. The argument is interpreted in degrees or radians according to the current mode setting. Use \(^\circ\) or \( \text{r} \) to override the mode setting.

\( (\text{matrix}) \)
Return the matrix tangent of square matrix, which is not the hyperbolic tangent of each element. Refer to \textit{cos()} for details. matrix must be diagonalizable and the result always contains floating-point numbers.

\textbf{tanh()} \( (\text{expr}), (\text{list}) \)
Return the hyperbolic tangent of the argument. The argument is interpreted in degrees or radians according to the current mode setting. Use \(^\circ\) or \( \text{r} \) to override the mode setting.

\( (\text{matrix}) \)
Return the matrix hyperbolic tangent of square matrix, which is not the hyperbolic tangent of each element. Refer to \textit{cos()} for details. matrix must be diagonalizable and the result always contains floating-point numbers.

\textbf{tanh}^{-1}()
\( (\text{expr}), (\text{list}) \)
Return the angle whose hyperbolic tangent is the argument. The result is returned as degrees or radians depending on the current Angle mode setting.

\( (\text{matrix}) \)
Return the matrix inverse hyperbolic tangent of square matrix, which is not the same as the inverse hyperbolic tangent of each element. Refer to \textit{cos()} for details. matrix must be diagonalizable and the result always contains floating-point numbers.

\textbf{taylor(}expr,\textbf{var,order[,point]})\textbf{)}
Return the Taylor polynomial for \( \text{expr} \) in \textit{var}. \textit{point} is the expansion point and defaults to zero. The polynomial includes non-zero terms of integer degree from 0 through \textit{order} in \textit{(var - point)}. \textit{taylor()} returns itself if there is not truncated power series of this \textit{order}, or negative or fractional exponents are required. Use substitution or temporary multiplication by a power of \textit{(var - point)} for a more general series.

\textbf{tCollect(}expr\textbf{)}
Return expression in which products and integer powers of sines and cosines are converted to a linear combination of sines and cosines of multiple arguments, angle sums and angle differences. Trigonometric polynomials are converted to linear combinations of harmonics. \textit{tCollect()} tends to reverse \textit{tExpand()}. Applying \textit{tExpand()} to a \textit{tCollect()} result, or vice versa, in two steps may simplify an expression.

\textbf{tExpand(}expr\textbf{)}
Return expression in which products and integer powers of sines and cosines are converted to a linear combination of sines and cosines of multiple arguments, angle sums and angle differences. Trigonometric polynomials are converted to linear combinations of harmonics. \textit{tExpand()} tends to reverse \textit{tExpand()}. Applying \textit{tExpand()} to a \textit{tCollect()} result, or vice versa, in two steps may simplify an expression.
differences are expanded. Best used in Radian mode, because degree-mode scaling interferes with expansion. tExpand() trends to reverse tCollect(). Applying tCollect() to a tExpand() result, or vice versa, in two steps may simplify an expression.

**Text string**
Display string in a dialog box. If used outside of a Dialog...EndDlog block, a dialog box is created, otherwise string is shown in the defined dialog box.

**Then**
See If.

**Title string[, label]**
Create the title of a menu or dialog box when used in a ToolBar or Custom structure, or a Dialog ... EndDlog block. *label* is only valid in the ToolBar structure. When used, it allows the menu choice to branch to *label*.

**tmpCnv(expr_°unit1,_°unit2)**
Return temperature value *expr* converted from *unit1* to *unit2*. Valid units are °C, °F, °K and °R. To convert a temperature range instead of a value, use ΔtmpCnv().

**ΔtmpCnv(expr_°unit1,_°unit2)**
Return temperature range *expr* converted from *unit1* to *unit2*. Valid units are °C, °F, °K and °R. To convert a temperature value instead of a range, use tmpCnv().

**Toolbar**
ToolBar : block : EndTBar
Create a toolbar menu. *block* statements can be either Title or Item. Item statements must have labels. Title statements must have labels if there are no Item statements.

**Trace**
Draw a SmartGraph and place the trace cursor on the first Y= function, at the previous cursor position, or at the reset position if regraphing was necessary. Allows operation of the cursor keys and most coordinate editing keys. Press [ENTER] to resume operation.

**Try**
Try : block1 : Else : block2 : EndTry
Execute *block1* until an error occurs, then transfer execution to *block2*. *errnum* contains the error number. See also ClrErr and PassErr.

**TwoVar xList,yList[][,freqList][,catList][,incList]**
Calculate two-variable statistics and update system statistics variables. All lists must have equal dimension except *incList*. The first four arguments must be variable names of c1 - c99. *incList* need not be a variable name and cannot be c1 - c99.

**Unarchiv var1[,var2][,var3]...**
Move argument variables from user data archive memory (flash) to RAM. Archived variables can be accessed, but not deleted, renamed or stored to, since the variable is locked. Use Archive to archive variables.

**unitV(vector)**
Return row- or column-unit vector, depending on the form of *vector*, which must be a single-row or single-column matrix.

**Unlock var1[,var2][,var3]...**
Unlock the argument variables. Use Lock to lock variables.

**variance(list[,freqList])**
Return the population variance of the elements of *list*. *freqList* elements specify the frequency of the corresponding elements of *list*. Both lists must have at least two elements, and the same number of elements.

**(matrix[,freqMatrix])**
Return row vector of the population variance of the column elements of *matrix*. *freqMatrix* elements specify the frequency of corresponding *matrix* elements. *matrix* must have at least two rows.

**when()**
*(condition,trueRes[,falseRes][,unknownRes])* Return *trueRes*, *falseRes*, or *unknownRes*, when *condition* is true, false or unknown. Returns the arguments when there are too few arguments to determine the result. Omit both *falseRes* and *unknownRes* to define an expression only when *condition* is true. *undef* can be used as a result.

**While**
While *condition* : block : EndWhile
Execute block as long as condition is true.

"With" See |

xor
exp1 xor exp2
list1 xor list2
matrix 1 xor matrix 2
Return true, false or a simplified form.

integer1 and integer2
Bit-by-bit 32-bit integer logical exclusive-or. Arguments larger than 32-bit signed values are reduced with symmetric modulo operation. Mode must be Auto or Exact.

XorPic picVar[,row][,column]
Display Graph screen and logically exclusive-or Graph screen pixels with those of picVar picture variable. Only pixels exclusive to the screen or picVar are turned on. row and column specify the pixel coordinates for the upper left corner of picVar; defaults are 0,0.

zeros()
(expr,var)
Return list of candidate values for var which make expr = 0. Equivalent to
exp→list(solve(expr=0,var))
zeros() cannot express implicit solutions, solutions which require inequalities or solutions which do not involve var.

({expr1,expr2,...},{var1,var2,...})
Return matrix of candidate solutions of simultaneous algebraic expressions expr in var. var may be a variable or a solution guess in the form var = guess. Each matrix row represents an alternate solution, with the variables ordered as in the {var} list.

If all equations are polynomials and you supply no guesses, zeros() uses the lexical Gröbner/Buchberger elimination to attempt to find all solutions. Simultaneous polynomial equations can have extra variables with no values. Solution variables of no interest may be omitted. Solutions may include arbitrary constants of the form @k, where k is an integer from 1 to 255. Computation time or memory exhaustion may depend on the order of the vars in the equations or variables list.

zeros() attempts to find all real solutions with Gaussian elimination if you include no guesses, any equation is in non-polynomial in any variable, but all equations are linear in the solution variables.

zeros() attempts to find one real solution (with an interactive approximate method) if the system is neither polynomial in all its variables nor linear in its solution variables. The number of solution variables must equal the number of equations and all other variables must simplify to numbers. Each solution variable starts at its guess value, or 0.0 if a guess is not used. Guesses may need to be close to the solution for convergence.

ZoomBox
Display the Graph screen, pauses so that a box can be drawn to define the new view window, then updates the window.

ZoomData
Adjust the window settings based on the current data plots and function graphs so that all data points are sampled, then displays the Graph Screen. Does not adjust ymin and ymax for histograms.

ZoomDec
Set x and y = 0.1 and displays the Graph screen with the origin centered.

ZoomFit
Display the Graph screen and set the dependent variable Window dimensions so that all the picture is displayed for the current independent variable settings.

ZoomIn
Display the Graph screen, pauses so that a center point can be set to zoom in, then updates the viewing window. The zoom magnitude depends on the Zoom factors xFact and yFact in 2D graph modes, and on xFact, yFact and zFact in 3D mode.

ZoomInt
Display the Graph screen, pauses so that a zoom center point can be set, adjusts the Window settings such that each pixel is an integer in all directions, then updates the viewing window.
ZoomOut
Display the Graph screen, pauses so that a zoom center point can be set, then updates the viewing window. The zoom magnitude depends on the Zoom factors $x_{\text{Fact}}$ and $y_{\text{Fact}}$ in 2D graph modes, and on $x_{\text{Fact}}$, $y_{\text{Fact}}$ and $z_{\text{Fact}}$ in 3D mode.

ZoomPrev
Display the Graph screen and update the viewing window with the settings before the previous zoom.

ZoomRcl
Display the Graph screen and update the viewing window with the settings stored with ZoomSto.

ZoomSqr
Display the Graph screen and adjust the x or y window settings so that each pixel represents an equal width and height in the coordinate system, then update the viewing window. In 3D mode, lengthen the shortest two axes to equal the longest axis.

ZoomStd
Set the Window variables to the following standard values, then update the viewing window.

Function graphing:
- $x: [-10, 10, 1], y: [-10, 10, 1], xres = 2$

Parametric graphing:
- $t: [0, 2\pi, \pi/24], x: [-10, 10, 1], y: [-10, 10, 1]$

Polar graphing:
- $\theta: [0, 2\pi, \pi/24], x: [-10, 10, 1], y: [-10, 10, 1]$

Sequence graphing:
- $x: [-10, 10, 1], y: [-10, 10, 1]$
  - $nmin = 1, nmax = 10, plotStrt = 1, plotStep = 1$

3D graphing:
- $x: [-10, 10, 14], y: [-10, 10, 14], z: [-10, 10]$  
  - $\text{eye}_{\theta} = 20, \text{eye}_{\phi} = 70, \text{eye}_{\psi} = 0, ncontour = 5$

Differential equations graphing:
- $t: [0, 10, 0], x: [-10, 10, 1], y: [-10, 10, 1]$
  - $ncurves = 0, Estep = 1, difftol = .001, fldres = 14, dtime = 0$

ZoomSto
Store the current Window settings in Zoom memory. Use ZoomRcl to restore the settings.

ZoomTrig
Display the Graph screen, set $x = \pi/24$, $x_{\text{scl}} = \pi/2$, center the origin, set the y settings to $[-4, 4, .5]$ and update the viewing window.

+ (add)

- $expr1 + expr2$
- Return the sum of $expr1$ and $expr2$

- $list1 + list2$
- $matrix1 + matrix2$
- Return list or matrix whose elements are the sum of the corresponding elements. Argument dimensions must be equal.

- $expr + list$
- $list + expr$
- Return list whose elements are the sum of the list elements and $expr$.

- $expr + matrix$
- $matrix + expr$
- Return matrix with $expr$ added to each diagonal element of square $matrix$. Use .+ (dot plus) to add an expression to each element.

- (subtract)

- $expr1 - expr2$
- Return $expr1 - expr2$

- $list1 - list2$
- $matrix1 - matrix2$
- Return list or matrix whose elements are the elements of $list2$ (or $matrix2$) subtracted from the corresponding elements of $list1$ (or $matrix1$). Argument dimensions must be equal.

- $expr - list$
- $list - expr$
- Return list where each element is each $list$ element subtracted from $expr$, or $expr$ subtracted from each list element.

- $expr - matrix$
- Return matrix: $(expr * identityMatrix) - matrix$
- $matrix$ must be square.

- $matrix - expr$
- Return matrix: $matrix - (expr * identityMatrix)$
- $matrix$ must be square.

(Note: use .- (dot minus) to subtract an expression from each element.)
* (multiply)
expr1 * expr2
Return the product of expr1 and expr2

list1 * list2
Return list whose elements are the products of the corresponding elements of list1 and list2. List dimensions must be equal.

matrix1 * matrix2
Return the matrix product of matrix1 and matrix2. The number of rows of matrix1 must equal the number of columns of matrix2.

expr * list
list * expr
Return list where each element is each list element multiplied by expr.

expr * matrix
matrix * expr
Return matrix whose elements are the product of expr and each element of matrix. This is the same as .* (dot multiply).

/ (divide)
expr1 / expr2
Return the quotient of expr1 divided by expr2

list1 / list2
Return list whose elements are the quotients of the corresponding elements of list1 and list2. List dimensions must be equal.

expr / list
list / expr
Return list where each element is the quotient of expr divided by each list element, or each list element divided by expr.

matrix / expr
Return matrix whose elements are the quotient of each element of matrix divided by expr. Use ./ (dot divide) to divide an expression by each matrix element.

- (negate)
-expr
-list
-matrix
Return the negation of the argument. If the argument is a binary or hexadecimal integer, return the two's-complement.

% (percent)
expr%
lis%
matri%
Return (argument/100)

= (equal)
expr1 = expr2
list1 = list2
matrix1 = matrix2
Return True if first argument can be determined to be equal to second argument. Return False if first argument cannot be determined to be equal to second argument. Otherwise, return a simplified form of the equations. For list and matrix arguments, return element-by-element comparisons.

# (not equal)
expr1 ≠ expr2
list1 ≠ list2
matrix1 ≠ matrix2
Return True if first argument can be determined to be not equal to second argument. Return False if first argument can be determined to be equal to second argument. Otherwise, return a simplified form of the inequality. For list and matrix arguments, return element-by-element comparisons.

< (less than)
expr1 < expr2
list1 < list2
matrix1 < matrix2
Return True if first argument can be determined to be less than second argument. Return False if first argument can be determined to be greater than or equal to the second argument. Otherwise, return a simplified form of the inequality. For list and matrix arguments, return element-by-element comparisons.

≤ (less than or equal to)
expr1 ≤ expr2
list1 ≤ list2
matrix1 ≤ matrix2
Return True if first argument can be determined to be less or equal to the second argument. Return False if first argument can be determined to be greater than the second argument. Otherwise, return a simplified form of the comparison. For list and matrix arguments, return element-by-element comparisons.
\texttt{> (greater than)}
\begin{verbatim}
expr1 > expr2
list1 > list2
matrix1 > matrix2
\end{verbatim}
Return True if first argument can be determined to be greater than the second argument. Return False if first argument can be determined to be less than or equal to the second argument. Otherwise, return a simplified form of the comparison. For list and matrix arguments, return element-by-element comparisons.

\texttt{≥ (greater than or equal to)}
\begin{verbatim}
expr1 ≥ expr2
list1 ≥ list2
matrix1 ≥ matrix2
\end{verbatim}
Return True if first argument can be determined to be greater than or equal to the second argument. Return False if first argument can be determined to be less than the second argument. Otherwise, return a simplified form of the comparison. For list and matrix arguments, return element-by-element comparisons.

\texttt{.+ (dot add)}
\begin{verbatim}
matrix1.+ matrix2
\end{verbatim}
Return matrix whose elements are the sums of the corresponding elements of \texttt{matrix1} and \texttt{matrix2}.

\texttt{expr.+ matrix}
\texttt{matrix .+ expr}
Return matrix whose elements are the sums of \texttt{expr} and \texttt{matrix}.

\texttt{- (dot subtract)}
\begin{verbatim}
matrix1.- matrix2
\end{verbatim}
Return matrix whose elements are the corresponding elements of \texttt{matrix2} subtracted from \texttt{matrix1}.

\texttt{expr.- matrix}
Return matrix whose elements are the elements of \texttt{matrix} subtracted from \texttt{expr}.

\texttt{matrix.- expr}
Return matrix whose elements are \texttt{expr} subtracted from the elements of \texttt{matrix}.

\texttt{* (dot multiply)}
\begin{verbatim}
matrix1 .* matrix2
\end{verbatim}
Return matrix whose elements are the products of each corresponding elements of \texttt{matrix1} and \texttt{matrix2}.

\texttt{expr .* matrix}
\texttt{matrix .* expr}
Return matrix whose elements are the product of \texttt{expr} and the elements of \texttt{matrix}.

\texttt{/ (dot divide)}
\begin{verbatim}
matrix1 .* matrix2
\end{verbatim}
Return matrix whose elements are the quotients of the corresponding elements of \texttt{matrix1} divided by \texttt{matrix2}.

\texttt{expr .* matrix}
\texttt{matrix .* expr}
Return matrix whose elements are the quotients of \texttt{expr} divided by the elements of \texttt{matrix}, or the elements of \texttt{matrix} divided by \texttt{expr}.

\texttt{^ (dot power)}
\begin{verbatim}
matrix1 .^ matrix2
\end{verbatim}
Return matrix whose elements are the elements of \texttt{matrix1} raised to the power of the corresponding \texttt{matrix2} elements.

\texttt{expr .^ matrix}
\texttt{matrix .^ expr}
Return matrix whose elements are \texttt{expr} raised to the power of each \texttt{matrix} element, or the elements of \texttt{matrix} raised to the \texttt{expr} power.

\texttt{! (factorial)}
\begin{verbatim}
expr!
list!
matrix!
\end{verbatim}
Return the factorial of the argument. Only returns a numeric value for non-negative integer arguments.

\texttt{& (append)}
\begin{verbatim}
string1 & string2
\end{verbatim}
Return string which is \texttt{string2} appended to \texttt{string1}.

\texttt{f() (integrate)}
\begin{verbatim}
(expr.var[,lower][,upper])
(list.var[,lower][,upper])
(matrix.var[,lower][,upper])
\end{verbatim}
Return the integral of the first argument with respect to \texttt{var}, from \texttt{lower} to \texttt{upper} integration limits. Return anti-derivative if \texttt{lower} and \texttt{upper} are omitted; constant of integration is omitted, but \texttt{lower} will be added as a constant of integration if \texttt{upper} is not used. Anti-derivatives may differ by a numeric constant. Piece-wise
constants may be added so that the anti-derivative is valid over a larger interval.

\( f() \) can be nested for multiple integrals, and the integration limits can depend on integration variables outside the limits.

When both lower and upper are present, \( f() \) attempts to subdivide the integral at discontinuities. Numerical integration is used in Auto mode when a symbolic anti-derivative cannot be determined. Numerical integration is tried first in Approx mode, but anti-derivatives are sought if numerical differentiation is inapplicable or fails.

\( f() \) returns itself for pieces of expr it cannot determine.

\( \sqrt() \) (square root)
\( \sqrt{\text{expr}}, \sqrt{\text{list}} \)
Return the square root of the argument.

\( \Pi(\text{expr}, \text{var}, \text{low}, \text{high}) \) (product)
Evaluate \( \text{expr} \) for each value of \( \text{var} \) from \( \text{low} \) to \( \text{high} \); return the product of the results. Return 1 if \( \text{high} = \text{low} - 1 \). If \( \text{high} < \text{low} - 1 \), return \( 1 / \Pi(\text{expr}, \text{var}, \text{high}+1, \text{low}-1) \)

\( \Sigma(\text{expr}, \text{var}, \text{low}, \text{high}) \) (sum)
Evaluate \( \text{expr} \) for each value of \( \text{var} \) from \( \text{low} \) to \( \text{high} \) and return the sum of the results. Return zero if \( \text{high} = \text{low} - 1 \). If \( \text{high} < \text{low} - 1 \), return \(-\Sigma(\text{expr}, \text{var}, \text{high}+1, \text{low}-1) \)

\( ^{\text{power}} \)
\( \text{expr}^{\text{expr2}} \)
Return \( \text{expr} \) raised to the power of \( \text{expr2} \).

\( \text{list1}^{\text{list2}} \)
Return list whose elements are the elements of \( \text{list1} \) raised to the powers of the corresponding elements of \( \text{list2} \).

\( \text{expr}^{\text{list}} \)
Return list whose elements are \( \text{expr} \) raised to the powers of the elements of \( \text{list} \).

\( \text{list}^{\text{expr}} \)
Return list whose elements are the elements of \( \text{list} \) raised to the \( \text{expr} \) power.

\( \text{matrix}^{\text{integer}} \)
Return \( \text{matrix} \) raised to the \( \text{integer} \) power. \( \text{matrix} \) must be square. If \( \text{integer} = -1 \), return the matrix inverse. If \( \text{integer} < -1 \), return the inverse matrix raised to the \(-\text{integer} \) power.

\# (indirection)
\# \text{varNameString}
Refer to the variable whose name is \text{varNameString}.

\( ^{\text{radian}} \)
\( \text{expr}^{\text{r}}, \text{list}^{\text{r}}, \text{matrix}^{\text{r}} \)
In Degree mode, multiply the argument by \( 180/\pi \). Return argument unchanged in Radian mode. Use \( ^{\text{r}} \) to force radians regardless of the current Angle mode.

\( ^{\text{degree}} \)
\( \text{expr}^{\circ}, \text{list}^{\circ}, \text{matrix}^{\circ} \)
In Radian mode, multiply the argument by \( \pi/180 \). Return argument unchanged in Degree mode. Use \( ^{\circ} \) to force radians regardless of the current Angle mode.

\( ^{\text{angle}} \)
\[\text{r}, \theta \_\text{angle} \]
\[\text{r}, \theta \_\text{angle}, \text{z} \]
\[\text{r}, \theta \_\text{angle}, \phi \_\text{angle} \]
Return argument coordinates as a vector depending on the current Vector format setting: rectangular, cylindrical or spherical

\( ^{\text{angle_angle}} \)
\[\text{r}, \angle \_\text{angle} \]
Enter a complex value in \( (r \angle \theta) \) format. \( \text{angle} \) is interpreted according to the current Angle mode setting.

\( ^{\text{degree}, \text{minute}, \text{second}} \)
\( \text{dd}^{\prime} \text{mm}^{\prime\prime} \text{ss}^{.\prime\prime} \)
Return \( \text{dd} + (\text{mm}/60) + (\text{ss}.	ext{ss}/3600) \), where \( \text{dd} \) may be positive or negative, \( \text{mm} \) and \( \text{ss} \) are non-negative numbers. Enter a number in base-60 format, which allows entering angle in degrees, minutes and seconds, regardless of the current Angle mode. Also allows entry of times as hours, minutes and seconds.

\( ^{\prime} \) (prime)
\( \text{var}^{\prime} \text{var}^{\prime\prime} \)
Enter prime symbol in a differential equation (DE). A single prime denotes a first-order DE, two prime symbols denote a second-order DE.
Designate the units for expr. All unit names begin with the underscore.

\[ \text{var}_. \]
Treat var as complex if it has no value. By default, variables without the underscore are treated as real. Variables with values are treated as the value type; real or complex. Complex numbers can be stored in variables without the underscore, but complex operations work better if underscores are used.

\[ \text{convert} \]
\[ \text{expr}_\text{unit1} \rightarrow \text{expr}_\text{unit2} \]
Convert expr with units unit1 to units unit2, excluding temperature conversions. The units must be in the same category. Use tmpCnv() and ΔtmpCnv() for temperature conversions.

\[ \text{10}^{()} \]
\( (\text{expr}), (\text{list}) \)
Return 10 raised to the power of the argument.

\[ (\text{matrix}) \]
Return 10 raised to the power of square matrix. This is not the same as 10 raised to the power of each element. Refer to cos() for calculation details. matrix must be diagonalizable, and the result always contains floating-point numbers.

\[ x^{-1} \] \( (^{^{-1}}) \)
\[ \text{expr}^{^{-1}} \]
\[ \text{list}^{^{-1}} \]
Return the reciprocal of the argument.

\[ \text{matrix}^{^{-1}} \]
Return the inverse of square matrix, which must be non-singular.

\[ \mid (\text{"with"}) \]
\[ \text{expr}|\text{Boolean1} [\text{and} \text{Boolean2}]...[\text{and} \text{BooleanN}] \]
Evaluate expr subject to the Boolean constraints. This allows substitutions, interval constraints and exclusions. If Boolean is an equality such as \( \text{var} = \text{value} \), then each occurrence of \( \text{var} \) in expr is substituted with value. A constraint is formed with two or more Booleans joined with ‘and’, where each Boolean is an inequality. An exclusion is a Boolean of the form \( \text{var} \neq \text{value} \), which is primarily used to exclude exact solutions of cSolve(), cZeros(), solve(), etc.

\[ \rightarrow (\text{store}) \]
\[ \text{expr}\rightarrow\text{var} \]
\[ \text{list}\rightarrow\text{var} \]
\[ \text{matrix}\rightarrow\text{var} \]
Create var if it does not exist and initialize it to expr, list or matrix. If var exists and is not locked or protected, replace its contents with expr, list or matrix.

\[ \text{expr}\rightarrow\text{functionName}([\text{parameter1},...]) \]
Create functionName if it does not exist and initialize it to expr. If functionName exists and is not locked or protected, replace its contents with expr. parameters are optional function arguments.

\[ \text{list}\rightarrow\text{functionName}([\text{parameter1},...]) \]
\[ \text{matrix}\rightarrow\text{functionName}([\text{parameter1},...]) \]
Create functionName if it does not exist and initialize it to list or matrix. If functionName exists and is not locked or protected, replace its contents with list or matrix. parameters are optional function arguments.

\[ \copyright (\text{comment}) \]
\[ \text{[text]} \]
Process text as a comment line in a program or function. \( \copyright \) may be at the beginning of a line or anywhere within the line. All text to the right of \( \copyright \) to the end of the line is the comment.

\[ \text{0b, 0h} \]
\[ \text{0bbinaryNumber} \]
\[ \text{0hhexadecimalNumber} \]
Denote a binary or hexadecimal integer. 0b or 0h must be entered regardless of the Base mode setting, otherwise the number is treated as decimal (base 10).
Reserved system variable names

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δtbl</td>
<td>ymax</td>
</tr>
<tr>
<td>Δx</td>
<td>nmax</td>
</tr>
<tr>
<td>Δy</td>
<td>nmin</td>
</tr>
<tr>
<td>Σx</td>
<td>nStat</td>
</tr>
<tr>
<td>Σy</td>
<td>ncontour</td>
</tr>
<tr>
<td>Σx²</td>
<td>plotStep</td>
</tr>
<tr>
<td>Σy²</td>
<td>plotStrt</td>
</tr>
<tr>
<td>Σy</td>
<td>q1</td>
</tr>
<tr>
<td>Σy²</td>
<td>q3</td>
</tr>
<tr>
<td>ox</td>
<td>r²</td>
</tr>
<tr>
<td>oy</td>
<td>r1()-r99()</td>
</tr>
<tr>
<td>θc</td>
<td>rc</td>
</tr>
<tr>
<td>θmax</td>
<td>regCoeff</td>
</tr>
<tr>
<td>θmin</td>
<td>regEq(x)</td>
</tr>
<tr>
<td>@step</td>
<td>seed1</td>
</tr>
<tr>
<td>c1-c99</td>
<td>seed2</td>
</tr>
<tr>
<td>corr</td>
<td>Sx</td>
</tr>
<tr>
<td>difto1</td>
<td>Sy</td>
</tr>
<tr>
<td>dtime</td>
<td>sysData</td>
</tr>
<tr>
<td>eqn</td>
<td>sysMath</td>
</tr>
<tr>
<td>errornum</td>
<td>tØ</td>
</tr>
<tr>
<td>estep</td>
<td>tblInput</td>
</tr>
<tr>
<td>exp</td>
<td>tابلStart</td>
</tr>
<tr>
<td>eyeθ</td>
<td>tc</td>
</tr>
<tr>
<td>eyeφ</td>
<td>tmax</td>
</tr>
<tr>
<td>eyε</td>
<td>tmin</td>
</tr>
<tr>
<td>f1dpc</td>
<td>tplot</td>
</tr>
<tr>
<td>f1dres</td>
<td>zstep</td>
</tr>
<tr>
<td>main</td>
<td>u1()-u99()</td>
</tr>
<tr>
<td>maxX</td>
<td>ztstep</td>
</tr>
<tr>
<td>maxY</td>
<td>xc</td>
</tr>
<tr>
<td>medStat</td>
<td>xfact</td>
</tr>
<tr>
<td>medx1</td>
<td>xmax</td>
</tr>
<tr>
<td>medx2</td>
<td>xgrid</td>
</tr>
<tr>
<td>medx3</td>
<td>xmax</td>
</tr>
<tr>
<td>medy1</td>
<td>xmin</td>
</tr>
<tr>
<td>medy2</td>
<td>xres</td>
</tr>
<tr>
<td>medy3</td>
<td>xsc1</td>
</tr>
<tr>
<td>minX</td>
<td>xt1()-xt99()</td>
</tr>
<tr>
<td>minY</td>
<td>y</td>
</tr>
<tr>
<td>nc</td>
<td>y1'-()-y99'()</td>
</tr>
<tr>
<td>ncontour</td>
<td>y1()-y99()'</td>
</tr>
<tr>
<td>ncurves</td>
<td>yc</td>
</tr>
</tbody>
</table>

EOS (Equation Operating System) Hierarchy

Exponentiation (^) and element-by-element exponentiation (.^) are evaluated from right to left, for example, 2^3^4 = 2^(3^4).

Post operations and exponentiation are performed before negation, for example, -9^2 = -(9^2).

Other operations are performed according the these priorities:

1. Parentheses (), brackets [] and braces {}
2. Indirection (#)
3. Function calls
4. Post operators: degrees-minutes-seconds (°,',")`, factorial (!), percentage (%), radian (*), transpose (^)
5. Exponentiation, power operator (^)
6. Negation (-)
7. String concatenation (&)
8. Multiplication (*), division (/)
9. Addition (+), subtraction (-)
10. Equality relations: =, ≠, <, ≤, >, ≥
11. Logical 'not'
12. Logical 'and'
13. Logical 'or', 'xor'
14. Constraint "with" operator (|)
15. Store (→)