

### **Entering real numbers**

Real numbers may contain a decimal point or the letter “e” to signify an “exponent of 10”. Use control-minus to enter a negative sign.

Examples: 1, 1.2, 1.2e12, -1.2e-12.

Real numbers are displayed using the current display mode: std, fix, or sci.

**Note:** Previous versions of RPN Calculator used shift-minus (underscore) for a negative sign. This has changed to allow units to be entered.

## Entering integers

Integers start with a "#". Integers may end with an 'b', 'o', 'd', or 'h', for binary, octal, decimal, or hexadecimal, respectively. If the type specifier is left off the current integer display mode is assumed.

Examples: #10o, #12d, #F0h, #F0.

Integers are displayed using the current base mode: bin, oct, dec, or hex.

Only the number of bits specified by the integer word size are relevant. Use stws to store the integer word size and rcws to recall the integer word size.

## **Entering variables**

Variables start and end with a single quote (').

Examples: 'a', 'b'.

## Entering vectors

Vectors start with an opening brace ([) and end with a closing brace (]). Each element must be a real number separated by spaces.

Example: [1 2 3]

Each element of a vector is displayed using the current display mode: std, fix, or sci.

## Entering matrices

Matrices start with an opening brace ([]) and end with a closing brace (]). Each row must also start with an opening brace and end with a closing brace. Each element must be a real number with columns separated by spaces.

Example: [[1 2][3 4]]

Each element of a matrix is displayed using the current display mode: std, fix, or sci.

## Entering lists

Lists start with an opening curly bracket ( { ) and end with a closing curly bracket ( } ). Elements can be of any type including lists. Elements are separated by spaces.

*Examples: {1 2}, {#1 {'a' 3}}*

## Entering programs

Programs start with two opening angle brackets (<<) and end with two closing angle brackets (>>). Elements of a program can be of any type including other programs. Elements are separated by spaces.

Examples: << 2 \* >>, << << 2 \* >> << 3 \* >> ifte >>

You can store a program in a variable and execute it by typing the name of the variable without the single quotes.

You can use the following programming structures in programs:

- >** Create local variables and use them in the following program  
Usage:           obj1 obj2 ... objn -> v1 v2 ... vn << ... >>
- do-until-end** Indefinite loop that gets executed at least once  
Usage:           **do** commands **until** condition **end**
- for-next** Definite loop incremented by one  
Usage:           begin end **for** index commands **next**
- for-step** Definite loop with a specified increment  
Usage:           begin end **for** index commands increment **step**
- if-then-[else]-end** Conditional structure  
Usage:           **if** condition **then** true-commands **end**  
                  **if** condition **then** true-commands **else** false-commands **end**
- iferr-then-[else]-end** Error trapping conditional structure  
Usage:           **iferr** commands **then** error-commands **end**  
                  **iferr** commands **then** error-commands **else** no-error-commands
- end**
- start-next** Definite loop that gets executed at least once and is incremented by one  
Usage:           begin end **start** index commands **next**
- start-step** Definite loop with a specified increment that gets executed at least once  
Usage:           begin end **start** index commands increment **step**
- while-repeat-end** Indefinite loop  
Usage:           **while** condition **repeat** commands **end**

Note: Do not use i (square-root of negative one) or any other built-in command for local variables or loop indices.

## Entering units

Numbers defined with specific units starts with a real number, have an underscore () in the middle, and end with a combination of base units, times (\*), divide (/), power (^), and real numbers.

Examples: 1\_m, -2.2E-3\_kg\*m^2/s^2.

Numbers with units are displayed using the current display mode: std, fix, or sci.

**Note:** Units are defined in an editable text file, units.txt.



## Entering complex numbers

Complex numbers start with are enclosed in parentheses ( ) and contain two real numbers, the real part and the imaginary part, separated by a comma (,).

Examples: (0,1), (1.2, 3.4).

Complex numbers are displayed using the current display mode: std, fix, or sci.

Complex numbers are displayed as length an angle in plr mode or as real part and imaginary part in rc† mode.

When in polar (plr) coordinate mode, the complex angle is displayed using the current angle mode: rad, deg, or grad.

**Legend**

**+ Add**

The + function returns the sum of two objects, as follows:

a b	->	c
a #b	->	#c
#a b	->	#c
#a #b	->	#c
[a] [b]	->	[c]
[[a]] [[b]]	->	[[c]]
{a} {b}	->	{a b}
{a} obj	->	{a obj}
obj {a}	->	{obj a}
a_u1 b_u2	->	c_u2
a (b)	->	(c)
(a) b	->	(c)
(a) (b)	->	(c)

**Legend**

**- Subtract**

The - function returns the difference of two objects, as follows:

a b	->	c
a #b	->	#c
#a b	->	#c
#a #b	->	#c
[a] [b]	->	[c]
[[a]] [[b]]	->	[[c]]
a_u1 b_u2	->	c_u2
a (b)	->	(c)
(a) b	->	(c)
(a) (b)	->	(c)

**Legend**

**\* Multiply**

The \* function returns the product of two objects, as follows:

a b	->	c
a #b	->	#c
#a b	->	#c
#a #b	->	#c
a [b]	->	[c]
[a] b	->	[c]
[[a]] [b]	->	[c]
a [[b]]	->	[[c]]
[[a]] b	->	[[c]]
[[a]] [[b]]	->	[[c]]
a_u1 b_u2	->	c_u3
a_u b	->	c_u
a b_u	->	c_u
a (b)	->	(c)
(a) b	->	(c)
(a) (b)	->	(c)

**Legend**

**/ Divide**

The / function returns the quotient of two objects, as follows:

a b	->	c
a #b	->	#c
#a b	->	#c
#a #b	->	#c
[a] b	->	[c]
[a] [[b]]	->	[c]
[[a]] b	->	[[c]]
[[a]] [[b]]	->	[[c]]
a_u1 b_u2	->	c_u3
a_u b	->	c_u
a b_u1	->	c_u2
a (b)	->	(c)
(a) b	->	(c)
(a) (b)	->	(c)

**Legend**

**^ Power**

The ^ function returns the object in the second position raised to the power of the object in the first position, as follows:

a b	->	c
a_u1 b	->	c_u2
a (b)	->	(c)
(a) b	->	(c)
(a) (b)	->	(c)

## Legend

### ! Factorial (Gamma)

The ! function returns the factorial of a real number if it is a positive integer. Otherwise, the ! function returns the gamma function of the number plus one, as follows:

a -> b

## Legend

### **== Equal Comparison**

The == function tests if the object in the second position is equal to the object in the first position, as follows:

a b	->	c
#a #b	->	#c

A true result is represented by the real number one. A false result is represented by the real number zero.



## Legend

### < Less Than Comparison

The < function tests if the object in the second position is less than the object in the first position, as follows:

a b	->	c
#a #b	->	#c

A true result is represented by the real number one. A false result is represented by the real number zero.

## Legend

### **<=**      **Less Than or Equal to Comparison**

The **<=** function tests if the object in the second position is less than or equal to the object in the first position, as follows:

a b	->	c
#a #b	->	#c

A true result is represented by the real number one. A false result is represented by the real number zero.

## Legend

### <> Not Equal Comparison

The <> function tests if the object in the second position is not equal to the object in the first position, as follows:

a b	->	c
#a #b	->	#c

A true result is represented by the real number one. A false result is represented by the real number zero.

## Legend

### > Greater Than Comparison

The > function tests if the object in the second position is greater than the object in the first position, as follows:

a b	->	c
#a #b	->	#c

A true result is represented by the real number one. A false result is represented by the real number zero.

## Legend

### **>=** Greater Than or Equal to Comparison

The **>=** function tests if the object in the second position is greater than or equal to the object in the first position, as follows:

a b	->	c
#a #b	->	#c

A true result is represented by the real number one. A false result is represented by the real number zero.

**Legend**

**% Percent**

The % function returns the product of two objects divided by 100, as follows:

- a b -> c
- a\_u b -> c\_u
- a b\_u -> c\_u

**Legend**

**abs            Absolute Value**

The **abs** function returns the absolute value of an object, as follows:

a	->	b
[a]	->	b
[[a]]	->	b
a_u	->	b_u
(a)	->	b

## Legend

### **acos**      **Arc Cosine**

The **acos** function returns the arc cosine of a real number, as follows:

a	->	b
(a)	->	(b)

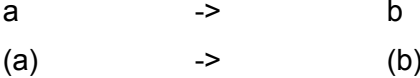
The angle returned depends on the current angle mode: rad, deg, or grad.



**Legend**

**acosh      Arc Hyperbolic Cosine**

The **acosh** function returns the arc hyperbolic cosine of a real number, as follows:



**Legend**

**alog**      **Base 10 Antilogarithm**

The **alog** function returns the base 10 antilogarithm of a real number, as follows:

$$\begin{array}{ccc} a & \rightarrow & b \\ (a) & \rightarrow & (b) \end{array}$$

**Legend**

**and Logical or Bitwise AND**

The **and** function returns the logical AND of two real numbers or the bitwise AND of two integers, as follows:

```
a b      ->      c
#a #b    ->      #c
```

A true result is represented by the real number one. A false result is represented by the real number zero.

## Legend

**arg**            **Complex Angle (Argument)**

The **arg** function returns the polar angle of a complex number, as follows:

(a)            ->            b

**Legend**

**->array      Create Matrix or Vector**

The **->array** function creates a matrix or vector, as follows:

$n_1 \dots n_a$  a      ->      [b]

$n_1 \dots n_{ab}$  {a b}      ->      [[c]]

## Legend

### **asin**      **Arc Sine**

The **asin** function returns the arc sine of a real number, as follows:

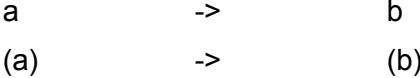
a	->	b
(a)	->	(b)

The angle returned depends on the current angle mode: rad, deg, or grad.

**Legend**

**asinh      Arc Hyperbolic Sine**

The **asinh** function returns the arc hyperbolic sine of a real number, as follows:



## Legend

### **atan**      **Arc Tangent**

The **atan** function returns the arc tangent of a real number, as follows:

a                      ->                      b  
(a)                     ->                     (b)

The angle returned depends on the current angle mode: rad, deg, or grad.



## Legend

**atanh**      **Arc Hyperbolic Tangent**

The **atanh** function returns the arc hyperbolic tangent of a real number, as follows:

a	->	b
(a)	->	(b)

## Legend

### **beep**      **Play a Note**

The **beep** function plays a note, as follows:

    frq dur      ->

The frequency is in Hertz. The duration is in seconds.

Note: **beep** does not work correctly under Win32s.

## Legend

### **bin**            **Set Binary Mode**

The **bin** function sets the integer base mode to binary, as follows:

->

All integers on the stack will be displayed in binary mode. For example, the integer 16 will be displayed as #10000b.

**Legend**

**b->r          Binary to Real Conversion**

The **b->r** function converts an integer to a real number, as follows:

$$\#a \quad \rightarrow \quad b$$

## Legend

### **calcfv**      **Calculate Future Value**

The **calcfv** function calculates the future value of money, as follows:

->                    a

The number of payments per year (stopyr) must be set before any time value of money calculations can be done. The default is monthly payments. Four out of five of the following must be set before calculating the fifth value: future value (stofv), annual interest rate (stoi %yr), number of payments (ston), payment value (stopmt), and present value (stopv).

## Legend

### **calci%yr      Calculate Annual Interest Rate**

The **calci%yr** function calculates the annual interest rate, as follows:

->                    a

This calculation is done using an iterative approach. The other values used in the calculation must be reasonable in order to get a reasonable result.

The number of payments per year (stopyr) must be set before any time value of money calculations can be done. The default is monthly payments. Four out of five of the following must be set before calculating the fifth value: future value (stofv), annual interest rate (stoi%yr), number of payments (ston), payment value (stopmt), and present value (stopv).

## Legend

### **calcn**      **Calculate Number of Payments**

The **calcn** function calculates the number of payments, as follows:

->                      a

The number of payments per year (stopyr) must be set before any time value of money calculations can be done. The default is monthly payments. Four out of five of the following must be set before calculating the fifth value: future value (stofv), annual interest rate (stoi %yr), number of payments (ston), payment value (stopmt), and present value (stopv).

## Legend

### **calcpmt      Calculate Payment**

The **calcpmt** function calculates the payment, as follows:

->                    a

The number of payments per year (stopyr) must be set before any time value of money calculations can be done. The default is monthly payments. Four out of five of the following must be set before calculating the fifth value: future value (stofv), annual interest rate (stoi %yr), number of payments (ston), payment value (stopmt), and present value (stopv).



## Legend

### **calcpv**      **Calculate Present Value**

The **calcpv** function calculates the present value of money, as follows:

->                      a

The number of payments per year (stopyr) must be set before any time value of money calculations can be done. The default is monthly payments. Four out of five of the following must be set before calculating the fifth value: future value (stofv), annual interest rate (stoi %yr), number of payments (ston), payment value (stopmt), and present value (stopv).

**Legend**

**ceil Ceiling**

The **ceil** function returns the next greater integer, as follows:

a	->	b
a_u	->	b_u

**Legend**

**%ch          Percent Change**

The **%ch** function returns the percent change of the object in the first position compared to the object in the second position, as follows:

a b                  ->                  c  
a\_u1 b\_u2        ->                  c

## Legend

**clear**      **Clear the Stack**

The **clear** function clears the entire stack, as follows:

obj<sub>1</sub> ... obj<sub>n</sub>    ->

## Legend

### **clsum**      **Clear Statistics Matrix**

The **clsum** function clears the statistics matrix, as follows:

->

## Legend

**clvar**      **Clear all Variables**

The **clvar** function clears all of the stored variables, as follows:

->

**Legend**

**cnrm          Column Norm**

The **cnrm** function returns the column norm of a matrix or vector, as follows:

[a]            ->          b  
[[a]]        ->          b

**Legend**

**comb      Combinations**

The **comb** function returns the number of combinations, as follows:

$$n \ r \quad \rightarrow \quad c$$



**Legend**

**con            Create Constant Matrix or Vector**

The **con** function returns a matrix or vector with all elements equal, as follows:

[a] b	->	[b]
{a} b	->	[c]
[[a]] b	->	[[b]]
{a b} c	->	[[d]]

**Legend**

**conj**            **Complex Conjugate**

The **conj** function returns conjugate of a complex number, as follows:

$$(a) \quad \rightarrow \quad (b)$$

## Legend

### **convert**      **Convert Units**

The **convert** function converts the object in the second position to the units of the object in the first position, as follows:

a\_u1 b\_u2      ->      c\_u2

**Legend**

**corr            Correlation Coefficient**

The **corr** function returns the correlation coefficient of X and Y in the statistics matrix, as follows:

->            a

**Legend**

**cos            Cosine**

The **cos** function returns the cosine an object, as follows:

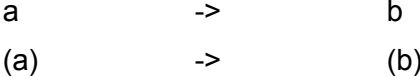
a	->	b
a_u	->	b
(a)	->	(b)

If an angle unit is not specified, the units of the input angle depends on the current angle mode: rad, deg, or grad.

**Legend**

**cosh      Hyperbolic Cosine**

The **cosh** function returns the hyperbolic cosine of a real number, as follows:



**Legend**

**cov            Covariance**

The **cov** function returns the covariance of X and Y in the statistics matrix, as follows:

->            a

**Legend**

**c->r          Complex to Real Conversion**

The **c->r** function returns the real part and the imaginary part of a complex number, as follows:

$$(a) \quad \rightarrow \quad r \ i$$



## Legend

### **cross**      **Cross Product**

The **cross** function returns the cross product of two three-element vectors, as follows:

[a] [b]      ->      [c]

## Legend

### **dot**      **Dot Product**

The **dot** function returns the dot product of two vectors, as follows:

$$[a] [b] \quad \rightarrow \quad c$$

**Legend**

**d->r          Degrees to Radians Conversion**

The **d->r** function converts a real number from degrees to radians, as follows:

$$a \quad \rightarrow \quad b$$

## Legend

### **dec**            **Set Decimal Mode**

The **dec** function sets the integer base mode to decimal, as follows:

->

All integers on the stack will be displayed in decimal mode. For example, the integer 16 will be displayed as #16d.

## Legend

### **deg**            **Set Degrees Mode**

The **deg** function sets the angle mode to degrees, as follows:

->

Trigonometric functions will use degree units, by default. The inverse trigonometric functions will return angles in degrees. There are 360 degrees in a full circle.

## Legend

### **det**            **Determinant of a Matrix**

The **det** function returns the determinant of a matrix, as follows:

`[[a]]`             $\rightarrow$             `b`

**Legend**

**depth**      **Depth of the Stack**

The **depth** function returns the depth of the stack, as follows:

->      a

**Legend**

**drop**            **Drop an Object**

The **drop** function drops an object from the stack, as follows:

obj            ->



## Legend

### **drop2**      **Drop two Objects**

The **drop2** function drops two objects from the stack, as follows:

obj<sub>1</sub> obj<sub>2</sub>      ->

## Legend

**dropn**      **Drop n Objects**

The **dropn** function drops n objects from the stack, as follows:

$obj_1 \dots obj_n \quad n \quad \rightarrow$

**Legend**

**dup            Duplicate an Object**

The **dup** function duplicates an object on the stack, as follows:

obj            ->            obj obj

**Legend**

**dup2 Duplicate two Objects**

The **dup2** function duplicates two objects on the stack, as follows:

obj1 obj2 -> obj1 obj2 obj1 obj2

## Legend

### **dupn Duplicate n Objects**

The **dupn** function duplicates n objects on the stack, as follows:

$\text{obj}_1 \dots \text{obj}_n \quad n \quad \rightarrow \quad \text{obj}_1 \dots \text{obj}_n \text{ obj}_1 \dots \text{obj}_n$

**Legend**

**e**                    **Constant e**

The **e** function returns 2.71828182845905, as follows:

->                    e

## Legend

### **eng**            **Set Engineering Mode**

The **eng** function sets the real number display mode to engineering notation with the given number of fixed digits, as follows:

n                    ->

All real numbers on the stack will be displayed in engineering mode. For example, the real number 12.345 will be displayed as 12.3E0 when fixed to 2 decimal places.

**Legend**

**eval            Evaluate an Object**

The **eval** function evaluates a program or variable, as follows:

obj            ->            depends



**Legend**

**exp            Natural Antilogarithm**

The **exp** function returns the natural antilogarithm of a real number, as follows:

a            ->            b  
(a)           ->            (b)

## Legend

### **fix**                    **Set Fixed Mode**

The **fix** function sets the real number display mode to fixed notation with the given number of fixed digits, as follows:

n                    ->

All real numbers on the stack will be displayed in fixed mode. For example, the real number 12.345 will be displayed as 12.34 when fixed to 2 decimal places. If the number is too big or too small to display in fixed mode, scientific mode is used instead.

**Legend**

**floor**      **Floor**

The **floor** function returns the next smaller integer, as follows:

a            ->      b  
a\_u         ->      b\_u

**Legend**

**fp                  Fractional Part**

The **fp** function returns the fractional part of an object, as follows:

a	->	b
a_u	->	b_u

**Legend**

**get            Get an Element**

The **get** function gets an element of a vector, matrix, or list, as follows:

- [a] b            ->            c
- [[a]] {b c}      ->            d
- {a} b            ->            obj

## Legend

### **grad**      **Set Grads Mode**

The **grad** function sets the angle mode to grads, as follows:

->

Trigonometric functions will use grad units, by default. The inverse trigonometric functions will return angles in grads. There are 400 grads in a full circle.

## Legend

### **hex**            **Set Hexadecimal Mode**

The **hex** function sets the integer base mode to hexadecimal, as follows:

->

All integers on the stack will be displayed in hexadecimal mode. For example, the integer 16 will be displayed as #10h.

**Legend**

**i                    Comlex Constant i**

The **i** function return the constant i, as follows:

->                    (i)



**Legend**

**idn**            **Identity Matrix**

The **idn** function creates an n x n identity matrix, as follows:

$$n \quad \rightarrow \quad [[a]]$$

**Legend**

**ift            If-Then**

The **ift** function evaluates the object in the first position if the condition in the second position is TRUE, as follows:

$b \text{ obj}_1 \rightarrow \text{depends}$

Any non-zero real number is TRUE. Zero is FALSE.

## Legend

### **ifte**            **If-Then-Else**

The **ifte** function evaluates the object in the second position if the condition in the third position is TRUE and evaluates the object in the first position otherwise, as follows:

$b \text{ obj}_T \text{ obj}_F \rightarrow \text{depends}$

Any non-zero real number is TRUE. Zero is FALSE.

**Legend**

**im**            **Imaginary Part**

The **im** function returns the imaginary part of a complex number, as follows:

$$(a) \quad \rightarrow \quad b$$

**Legend**

**inv            Inverse**

The **inv** function returns the inverse of an object, as follows:

a	->	b
[[a]]	->	[[b]]
a_u1	->	b_u2
(a)	->	(b)

**Legend**

**ip Integer Part**

The **ip** function returns the integer part of an object, as follows:

a	->	b
a_u	->	b_u

## Legend

### **->list          Create List**

The **->list** function creates a list of n objects, as follows:

$\text{obj}_1 \dots \text{obj}_n \quad \rightarrow \quad \{\text{obj}_1 \dots \text{obj}_n\}$

**Legend**

**In Natural Logarithm**

The **In** function returns the natural logarithm of a real number, as follows:

$$\begin{array}{ccc} a & \rightarrow & b \\ (a) & \rightarrow & (b) \end{array}$$



**Legend**

**log            Base 10 Logarithm**

The **log** function returns the base 10 logarithm of a real number, as follows:

$$\begin{array}{ccc} a & \rightarrow & b \\ (a) & \rightarrow & (b) \end{array}$$

**Legend**

**lr                    Linear Regression**

The **lr** function returns the intercept and slope using linear regression on the statistics matrix, as follows:

->                    b m

**Legend**

**mant**      **Mantissa**

The **mant** function returns the mantissa of a real number, as follows:

a                      ->                      b

**Legend**

**max**            **Maximum**

The **max** function returns the higher value object, as follows:

a b            ->            c  
a\_u1 b\_u2    ->            c\_u3

**Legend**

**maxr**            **Maximum Real**

The **maxr** function returns 1.79769313486232E+308, as follows:

->            a

## Legend

### **maxsum**      **Maximum Values**

The **maxsum** function returns the maximum values in each column of the statistics matrix, as follows:

->            [a]

## Legend

**mean**      **Mean**

The **mean** function returns the mean values of each column of the statistics matrix, as follows:

->      [a]

**Legend**

**min            Minimum**

The **min** function returns the lower value object, as follows:

a b            ->            c  
a\_u1 b\_u2    ->            c\_u3



**Legend**

**minr**            **Minimum Real**

The **minr** function returns 2.2250738585072E-308, as follows:

->            a

## Legend

### **minsum**      **Minimum Values**

The **minsum** function returns the minimum values in each column of the statistics matrix, as follows:

->            [a]

**Legend**

**mod**      **Modulo**

The **mod** function returns the modulo of two real numbers, as follows:

$$a \ b \quad \rightarrow \quad c$$

**Legend**

**neg Negate**

The **neg** function negates an object, as follows:

a	->	b
#a	->	#b
[a]	->	[b]
[[a]]	->	[[b]]
a_u	->	b_u
(a)	->	(b)

## Legend

### **not**            **Logical or Bitwise NOT**

The **not** function returns the logical NOT of a real number or the bitwise NOT of an integer, as follows:

a	->	b
#a	->	#b

A true result is represented by the real number one. A false result is represented by the real number zero.

**Legend**

**nsum**      **Number of Rows**

The **nsum** function returns the number of data points in the statistics matrix, as follows:

->      a

**Legend**

**obj->**            **Object Components**

The **obj->** function breaks an object into its components, as follows:

obj            ->            depends

## Legend

### **oct**            **Set Octal Mode**

The **oct** function sets the integer base mode to octal, as follows:

->

All integers on the stack will be displayed in octal mode. For example, the integer 16 will be displayed as #20o.



## Legend

**off**            **Exit RPN Calculator**

The **off** function turns RPN Calculator off, as follows:

->

## Legend

### **or**                    **Logical or Bitwise OR**

The **or** function returns the logical OR of two real numbers or the bitwise OR of two integers, as follows:

a b	->	c
#a #b	->	#c

A true result is represented by the real number one. A false result is represented by the real number zero.

**Legend**

**over Duplicate Level 2 Object**

The **over** function duplicates the object in the second position, as follows:

obj<sub>1</sub> obj<sub>2</sub> -> obj<sub>1</sub> obj<sub>2</sub> obj<sub>1</sub>

**Legend**

**perm          Permutations**

The **perm** function returns the number of permutations, as follows:

$$n \ r \quad \rightarrow \quad c$$

## Legend

**pi**            **Constant**

The **pi** function returns 3.141592653589793, as follows:

->

## Legend

### **pick Duplicate Level n+1 Object**

The **pick** function duplicates the object in position n+1, as follows:

$obj_1 \dots obj_n \quad \rightarrow \quad obj_1 \dots obj_n \quad obj_1$

## Legend

### **plr**            **Set Polar Mode**

The **plr** function sets the coordinate mode to polar, as follows:

->

**Legend**

**predx**      **Predicted X Value**

The **predx** function returns the predicted X value using the curve calculated by the lr function, as follows:

$$y \quad \rightarrow \quad x$$



**Legend**

**predy**      **Predicted Y Value**

The **predy** function returns the predicted Y value using the curve calculated by the lr function, as follows:

$$x \quad \rightarrow \quad y$$

**Legend**

**purge**      **Purge Variable**

The **purge** function deletes a variable, as follows:

'a'            ->

**Legend**

**put Put an Element**

The **put** function puts an element into a vector, matrix, or list, as follows:

```
[a] b c      ->      [d]
[[a]] {b c} d ->     [[e]]
{a} b obj    ->     {c}
```

## Legend

### **rad**            **Set Radians Mode**

The **rad** function sets the angle mode to radians, as follows:

->

Trigonometric functions will use radian units, by default. The inverse trigonometric functions will return angles in radians. There are  $2\pi$  radians in a full circle.

## Legend

**rand**            **Random Number**

The **rand** function generates a pseudo-random number, as follows:

->            a

The rdz function should be called once before using **rand**.

**Legend**

**r->b      Real to Binary Conversion**

The **r->b** function converts a real number to an integer, as follows:

$$a \quad \rightarrow \quad \#b$$

**Legend**

**r->c      Real to Complex Conversion**

The **r->c** function converts real part and the imaginary part of a complex number to a complex number, as follows:

$$r \ i \quad \rightarrow \quad (a)$$

**Legend**

**rcl**            **Recall Variable**

The **rcl** function recalls a variable, as follows:

          'a'            ->            obj



**Legend**

**rclfv**            **Recall Future Value**

The **rclfv** function recalls the future value of money, as follows:

->            a

**Legend**

**rcli%yr**      **Recall Annual Interest Rate**

The **rcli%yr** function recalls the annual interest rate, as follows:

->            a

**Legend**

**rcln**            **Recall Number of Payments**

The **rcln** function recalls the number of payments, as follows:

->            a

**Legend**

**rclpmt**      **Recall Payment**

The **rclpmt** function recalls the payment, as follows:

->      a

**Legend**

**rclpv**      **Recall Present Value**

The **rclpv** function recalls the present value, as follows:

->      a

**Legend**

**rclpyr**      **Recall Number of Payments per Year**

The **rclpyr** function recalls the number of payments per year, as follows:

->            a

## Legend

### **rclsum**      **Recall Statistics Matrix**

The **rclsum** function recalls the statistics matrix, as follows:

->            [[a]]

The matrix is not recalled if it contains more than 16 rows.

## Legend

### **rct**            **Set Rectangular Mode**

The **rct** function sets the coordinate mode to rectangular, as follows:

->



**Legend**

**rcws**            **Recall Integer Word Size**

The **rcws** function recalls the integer word size, as follows:

->            n

**Legend**

**r->d          Radians to Degrees Conversion**

The **r->d** function converts a real number from radian to degrees, as follows:

a                  ->                  b

## Legend

### **rdz**            **Randomize**

The **rdz** function sets the seed for generating pseudo-random numbers, as follows:

a                    ->

If zero is specified for the seed, the seed is based on the current time.

To generate pseudo-random numbers, use the rand function.

**Legend**

**re**            **Real Part**

The **re** function returns the real part of a complex number, as follows:

$$(a) \quad \rightarrow \quad b$$

**Legend**

**rl**            **Rotate Left one Bit**

The **rl** function rotates the bits of an integer one bit to the left, as follows:

**#a**            **->**            **#b**

**Legend**

**rlb**            **Rotate Left one Byte**

The **rlb** function rotates the bits of an integer one byte to the left, as follows:

#a            ->            #b

**Legend**

**rnd            Round**

The **rnd** function rounds the real part of an object off to the given number of decimal places, as follows:

a	->	b
[a]	->	[b]
[[a]]	->	[[b]]
a_u	->	b_u
(a)	->	(b)

**Legend**

**rnorm      Row Norm**

The **rnorm** function returns the row norm of a matrix or vector, as follows:

[a]            ->            b  
[[a]]          ->            b



**Legend**

**roll            Roll Up n Objects**

The **roll** function rolls n objects up, as follows:

$$\text{obj}_1 \dots \text{obj}_n \quad n \quad \rightarrow \quad \text{obj}_2 \dots \text{obj}_n \text{ obj}_1$$

**Legend**

**rolld            Roll Down n Objects**

The **rolld** function rolls n objects down, as follows:

$$\text{obj}_1 \dots \text{obj}_n \quad \rightarrow \quad \text{obj}_n \text{ obj}_1 \dots \text{obj}_{n-1}$$

**Legend**

**rot            Rotate three Objects**

The **rot** function rotates three objects, as follows:

obj<sub>1</sub> obj<sub>2</sub> obj<sub>3</sub> ->            obj<sub>2</sub> obj<sub>3</sub> obj<sub>1</sub>

**Legend**

**rr**            **Rotate Right one Bit**

The **rr** function rotates the bits of an integer one bit to the right, as follows:

**#a**            **->**            **#b**

## Legend

**rrb**            **Rotate Right one Byte**

The **rrb** function rotates the bits of an integer one byte to the right, as follows:

**#a**            **->**            **#b**

## Legend

**rsd**            **Residual Matrix or Vector**

The **rsd** function returns the correction to the solution of a system of equations, as follows:

$$\begin{array}{l} [b] \ [a] \ [z] \ \rightarrow \ [c] \\ [[b]] \ [[a]] \ [[z]] \ \rightarrow \ [[c]] \end{array}$$

## Legend

### **sci**            **Set Scientific Mode**

The **sci** function sets the real number display mode to scientific notation with the given number of fixed digits, as follows:

n                    ->

All real numbers on the stack will be displayed in scientific mode. For example, the real number 12.345 will be displayed as 1.23E1 when fixed to 2 decimal places.

## Legend

### **sdev**          **Standard Deviation**

The **sdev** function returns the standard deviation of each column of the statistics matrix, as follows:

->          [a]



**Legend**

**sign**            **Sign**

The **sign** function returns the sign of an object, as follows:

a	->	b
a_u	->	b_u
(a)	->	(b)

**Legend**

**sin**            **Sine**

The **sin** function returns the sine of an object, as follows:

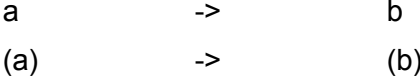
a	->	b
a_u	->	b
(a)	->	(b)

If an angle unit is not specified, the units of the input angle depends on the current angle mode: rad, deg, or grad.

**Legend**

**sinh          Hyperbolic Sine**

The **sinh** function returns the hyperbolic sine of a real number, as follows:



**Legend**

**size            Dimensions of an Object**

The **size** function returns the dimensions of an object, as follows:

[a]	->	{n}
[[a]]	->	{n m}
{a}	->	n
obj	->	1

**Legend**

**sl**                    **Shift Left one Bit**

The **sl** function shifts the bits of an integer one bit to the left, as follows:

**#a**                    **->**                    **#b**

**Legend**

**slb**            **Shift Left one Byte**

The **slb** function shifts the bits of an integer one byte to the left, as follows:

#a            ->            #b

**Legend**

**sq            Square**

The **sq** function returns the square of an object, as follows:

a	->	b
a_u1	->	b_u2
(a)	->	(b)

**Legend**

**sqrt      Square Root**

The **sqrt** function returns the square root of an object, as follows:

a	->	b
a	->	(b)
a_u1	->	b_u2



**Legend**

**sr**                    **Shift Right one Bit**

The **sr** function shifts the bits of an integer one bit to the right, as follows:

**#a**                    **->**                    **#b**

## Legend

**srb**            **Shift Right one Byte**

The **srb** function shifts the bits of an integer one byte to the right, as follows:

**#a**            **->**            **#b**

## Legend

### **std**            **Set Standard Mode**

The **std** function sets the real number display mode to standard notation, as follows:

->

All real numbers on the stack will be displayed in standard mode. For example, the real number 12.345 will be displayed as 12.345. If the number is too big or too small to display in standard mode, scientific mode is used instead.

## Legend

**sto**            **Store Object in a Variable**

The **sto** function stores an object in a variable, as follows:

obj 'a'            ->

**Legend**

**stofv**      **Store Future Value**

The **stofv** function stores the future value of money, as follows:

a                      ->

**Legend**

**stoi%yr      Store Annual Interest Rate**

The **stoi%yr** function stores the annual interest rate, as follows:

a                      ->

**Legend**

**ston**            **Store Number of Payments**

The **ston** function stores the number of payments, as follows:

a                    ->

**Legend**

**stopmt      Store Payment**

The **stopmt** function stores the payment, as follows:

a                      ->



**Legend**

**stopv          Store Present Value**

The **stopv** function stores the present value, as follows:

a                  ->

## Legend

**stopyr**      **Store Number of Payments per Year**

The **stopyr** function stores the number of payments per year, as follows:

a                      ->

## Legend

**stosum**      **Store Statistics Matrix**

The **stosum** function stores a matrix in the statistics matrix, as follows:

[[a]]      ->

**Legend**

**stws**            **Store Integer Word Size**

The **stws** function stores the integer word size, as follows:

n                    ->

**Legend**

**sum+      Add Row to Statistics Matrix**

The **sum+** function adds a row or rows to the statistics matrix, as follows:

- a            ->
- [a]          ->
- [[a]]       ->

## Legend

### **sum- Remove Row from Statistics Matrix**

The **sum-** function removes a row from the statistics matrix, as follows:

-> [a]

## Legend

### **sumpar**      **Statistics Matrix Parameters**

The **sumpar** function returns the statistics matrix parameters, as follows:

->                    {a}

The list contains five elements: selected X column (xcol), selected Y column (ycol), intercept, slope (lr), and curve-fitting model (linfit, expfit, logfit, pwrfit, bestfit). Currently, only the linear fit (linfit) model is available.

**Legend**

**sumx**      **Sum of X**

The **sumx** function returns the sum of the X column in the statistics matrix, as follows:

->      a



## Legend

**sumxx**      **Sum of X<sup>2</sup>**

The **sumxx** function returns the sum of the X column squared in the statistics matrix, as follows:

->            a

**Legend**

**sumxy**      **Sum of X\*Y**

The **sumxy** function returns the sum of the X column times the Y column in the statistics matrix, as follows:

->            a

**Legend**

**sumy**      **Sum of Y**

The **sumy** function returns the sum of the Y column in the statistics matrix, as follows:

->      a

**Legend**

**sumyy**      **Sum of Y^2**

The **sumyy** function returns the sum of the Y column squared in the statistics matrix, as follows:

->            a

## Legend

**swap**      **Swap two Objects**

The **swap** function swaps two objects, as follows:

obj<sub>1</sub> obj<sub>2</sub>      ->      obj<sub>2</sub> obj<sub>1</sub>

**Legend**

**%t            Percent of Total**

The **%t** function returns the percentage of the object in the first position compared to the object in the second position, as follows:

a b            ->            c  
a\_u1 b\_u2    ->            c

**Legend**

**tan            Tangent**

The **tan** function returns the tangent of an object, as follows:

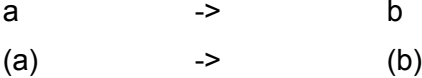
a	->	b
a_u	->	b
(a)	->	(b)

If an angle unit is not specified, the units of the input angle depends on the current angle mode: rad, deg, or grad.

**Legend**

**tanh          Hyperbolic Tangent**

The **tanh** function returns the hyperbolic tangent of an object, as follows:





**Legend**

**tot Totals**

The **tot** function returns the totals of each column of the statistics matrix, as follows:

-> [a]

**Legend**

**trn                    Transpose of a Matrix**

The **trn** function returns transpose of a matrix, as follows:

$$[[a]] \quad \rightarrow \quad [[b]]$$

**Legend**

**trnc            Truncate**

The **trnc** function truncates the real part of an object at the given number of decimal places, as follows:

a	->	b
[a]	->	[b]
[[a]]	->	[[b]]
a_u	->	b_u
(a)	->	(b)

**Legend**

**type            Type of an Object**

The **type** function returns a real number specifying the type of the given object, as follows:

obj            ->            n

The return value for each object type is:

Real	0
Complex	1
Vector	3
Matrix	3
List	5
Variable	6
Program	8
Integer	10
Unit	13

**Legend**

**ubase      Convert to SI Base Units**

The **ubase** function converts a unit object to SI base units, as follows:

$$a_{u1} \quad \rightarrow \quad b_{u2}$$

**Legend**

**->unit      Create Unit**

The **->unit** function creates a unit object from the real number in the second position and the units from the object in the first position, as follows:

$$a \ b\_u \quad \rightarrow \quad a\_u$$

**Legend**

**uval**            **Unit Value**

The **uval** function returns the real part of a unit object, as follows:

a\_u            ->            a

## Legend

**var**            **Variance**

The **var** function returns the variance of each column of the statistics matrix, as follows:

->            [a]



## Legend

### **vars**      **Get List of all Variables**

The **vars** function returns a list of all the variables, as follows:

->      {a}

**Legend**

**wait**            **Wait**

The **wait** function puts RPN Calculator to sleep for n seconds, as follows:

n                    ->

**Legend**

**xcol**            **Select X Column**

The **xcol** function selects the independent column of the statistics matrix, as follows:

x                    ->

## Legend

### **xor**            **Logical or Bitwise Exclusive OR**

The **xor** function returns the logical exclusive OR of two real numbers or the bitwise exclusive OR of two integers, as follows:

a b	->	c
#a #b	->	#c

A true result is represented by the real number one. A false result is represented by the real number zero.

**Legend**

**xpon**      **Exponent**

The **xpon** function returns the exponent of a real number, as follows:

$$a \quad \rightarrow \quad b$$

**Legend**

**xroot      Xth Root**

The **xroot** function returns the xth root of an object, as follows:

$$\begin{array}{l} y \ x \quad \rightarrow \quad a \\ y_{u1} \ x \quad \rightarrow \quad a_{u2} \end{array}$$

**Legend**

**ycol**            **Select Y Column**

The **ycol** function selects the dependent column of the statistics matrix, as follows:

x                    ->

**About Box** - Displays copyright and version information. Click OK to dismiss the dialog box.



**New** - Clears the stack, any RPN file and all stored information including custom programs and defined variables.

To clear the stack only, use the **clear** function.

**Open** - Opens a saved RPN file.

**Save** - Updates the open RPN file with current values for the stack, defined variables and programs.

If you are working with a new untitled file, clicking this command opens the Save As dialog where you can identify a filename and path.

**Save As** - Opens the Save As dialog where you can save all current values including the stack, defined variables and programs to an RPN file. You can create a new RPN file or select an existing file to be replaced.

**Recently used files** - Lists your four most recently used RPN files.

**Exit** - Closes the calculator. RPN Calculator prompts you to save if you have made any changes since the last save.

Edit menu

**Undo** - Restores the stack to a previous state. Currently, undo does not restore any modes or variables. Up to 100 levels of undo are currently available.



**Redo** - Restores the stack to a state before undo was selected. Currently, redo does not restore any modes or variables. Up to 100 levels of redo are currently available.

**Cut** - Cuts selected text from the entry field to the clipboard. This command does not work for objects on the stack.

This command is available only if you have selected text.

**Copy** - Copies selected text to the clipboard. You can select an object from the stack or you can select part or all of the information in the entry field.

This command is available only if you have selected text.

**Paste** - Pastes information from the clipboard to the current position on the entry field.

**Edit** - Places an object in the entry field for editing. If the object is the name of a global variable, the contents of the variable are placed in the entry field for editing.

## View menu

- keypad number keys are all aliased to hid\_keypad1
- all keys that have an equivalent tool are aliased to that tool

**Tool bar** - Shows/hides the toolbar. The toolbar provides easy access to file and editing functions and to help.

**Status bar** - Shows/hides the status bar.

The status bar displays a short description of all the tools on the tool menu.

The status bar also displays error messages when RPN Calculator cannot carry out a function you have entered.



**Key pad** - Shows/hides the keypad.

**Custom Tool Bar** - Shows/hides the custom tool bar. You can create a custom tool palette by creating buttons for tools, variables, programs or text.

**Number keys** - Inserts the number at the current cursor position in the entry field.

**Decimal point** - Inserts a decimal point at the current cursor position in the entry field.

**Letter keys** - Inserts the letter at the current cursor position in the entry field.

**Integer Symbol** - Inserts the integer symbol at the current cursor position in the entry field.

**Negative sign** - Insert a negative sign at the current cursor position in the entry field.

**Exponent key** - Insert the letter E to separate the exponent from the mantissa of a real number.



**Enter Key** - Adds the information in the entry field to the first position on the stack.

If the entry field is empty, clicking Enter will duplicate the object in the first position on the stack.

Help menu

**Help Topics** - Opens the Help dialog where you can access help through a table of contents, an index or a full text search.

**What's this?** - Provides mouse access to context sensitive help for RPN Calculator elements.

**About** - Provides version information about RPN Calculator.

**Angle Mode** - Sets the mode in which angles are represented: RAD (radians), DEG (degrees), GRAD (grads).

**Coordinate Mode** - Sets the mode in which complex numbers are represented: PLR (polar), RCT (rectangular).


**Integer Base Mode** - Sets the base mode for integers: BIN (binary), OCT (octal), DEC (decimal), HEX (hexadecimal).




**Display Mode** - Sets the real number display format: STD (standard), FIX (fixed notation), SCI (scientific notation).

**Stack** - Displays all the objects currently on the stack including numbers, variables and programs. If the list of objects on the stack exceed the stack display area, you can scroll up or down or see more. You can also copy objects from the stack into the entry field to edit them.

**Entry field** - Lets you add objects to the stack and execute commands.

Choose *Help | Help Topics* from the menu or press the  button on the tool bar to get more help.

Choose *Help | What's This?* from the menu or press the  button on the tool bar to get help on specific menu options or controls.

**Status Bar** - The status bar displays a short description of all the tools on the tool menu.

The status bar also displays error messages when RPN Calculator cannot carry out a function you have entered.

**Options Tab** - Lets you set RPN Calculator options.

*Error Beep* check box - Enables/disables an error beep when RPN Calculator cannot carry out a requested function.

*Automatically Save...* check box - Enables/disables automatic saving/quitting on exit. When this option is checked you will not be prompted to save on exit. An open file will automatically be saved. A new file will not be saved.

*Stay On Top* check box - Enables/disables staying on top when not active.

*Load...Positions* check box - Enables/disables the loading and saving of tool bar positions in the registry. Starting and exiting RPN Calculator will be faster when this option is disabled.

*Default Number of Fixed Digits* field - Lets you change the default number of fixed digits for real numbers. This affects fixed notation (FIX) and scientific notation (SCI) display modes when they are selected from the Display Mode drop down control.

**Custom Tool Bar Tab** - Lets you create a custom tool bar. You can create buttons for variables, programs and text strings.

*Command* field - Contains the name of a variable or program to add to the custom tool bar. The resulting button will execute the program or recall the variable.

*Text to Insert* field - Contains the text to be inserted into the entry field. The resulting button will insert the text into the entry field.

*Add Command* button - Adds the command in the *Command* field to the *Selected Buttons* list.

*Add Separator* button - Adds a space which separates buttons on the custom tool bar.

*Insert Text* button - Adds the text in the *Text to Insert* field to the *Selected Buttons* list.

*Remove Button* button - Removes the highlighted button from the *Selected Buttons* list.

*Selected Buttons* list - Lists all the buttons that will appear on the custom tool bar.

*Move Up/Down* buttons - Moves the highlighted button up or down in the *Selected Buttons* list.

**Options** - Opens the options dialog where you can create buttons for your custom tool bar. You can also set various RPN Calculator preferences.

**Execute Custom Command** - Executes the command displayed in the button's caption.



**Insert Custom Text** - Inserts the text displayed in the button's caption at the current cursor position in the entry field.

**Extended Entry Field Button** - Displays the extended entry field dialog.

**Extended Entry Field Dialog** - Allows more room for entering programs than the regular entry field.

**Insert global variable** - Inserts the specified global variable.

**Insert unit** - Inserts the symbol for the specified unit.

# - Inserts the symbol for an integer.

‘ ‘ - Inserts the symbol for a variable.

{ } - Inserts the symbol for a list.



[ ] - Inserts the symbol for a vector.

**[[ ]]** - Inserts the symbol for a matrix.

<< >> - Inserts the symbol for a program.

-> - Inserts the symbol for local variables.

( , ) - Inserts the symbol for a complex number.

\_ - Inserts the symbol for a unit.

**do** - Inserts symbol for do-until-end.

**else** - Inserts symbol for if[err]-then-else-end.



**end** - Inserts end keyword.

**for** - Inserts symbol for for-next/step.

**if** - Inserts symbol for if-then-[else]-end.

**iferr** - Inserts symbol for iferr-then-[else]-end.

**next** - Inserts symbol for for/start-next.

**repeat** - Inserts symbol for while-repeat-end.

**start** - Inserts symbol for start-next/step.

**step** - Inserts symbol for for/start-step.



**then** - Inserts symbol for if[err]-then-[else]-end.

**until** - Inserts symbol for do-until-end.

**while** - Inserts symbol for while-repeat-end.

## Saving programs and variables

1. Enter a program for the function or the information you want to save.
2. Assign the program to a variable.
3. Click File, Save to save to the current RPN file or File, Save As to save to a new RPN file.

See also:     Reviewing programs and variables  
              Copying programs and variables from one RPN file to another.

## Reviewing and editing programs and variables

1. Select vars from the Tools menu to display a list of all the variables on the stack.
2. Enter the variable you want to review.
3. Select rcl from the Tools menu to recall the program or information assigned to the variable to position 1 on the stack.
4. Select Edit from the Tools menu to move the information from position 1 on the stack to the entry field where you can make changes.
5. Change the program.
6. Identify a variable to which the program will be assigned.
7. Select sto from the tools menu.

## **RPN Calculator**

RPN Calculator is a reverse polish notation programmable calculator for science and engineering. It supports over 180 functions and 9 data types (real, integer, variable, vector, matrix, list, program, unit, and complex). The stack, variables, and programs can be saved and reloaded. The number of levels of stack that can be used is virtually unlimited. If the number of levels in the stack exceeds the display, you can scroll up or down to see more.

## **Disclaimer**

Tom Boldt accepts no responsibility for damages resulting from the use of the RPN Calculator and makes no warranty or representation, either express or implied, including but not limited to, any implied warranty of merchantability or fitness for a particular purpose. RPN Calculator is provided "as is," and you, its user, assume all risks when using it. Tom Boldt does not guarantee correct mathematical results.

Copyright © 1992-1997 Tom Boldt

**Installation**

The files rpncalc.exe, rpncalc.cnt, rpncalc.hlp, and units.txt must be unzipped to the same directory. Set up an icon or shortcut to rpncalc.exe. A command line argument of a saved filename is optional (Note: to use this feature, the saved file must first be created with RPN Calculator).

Registered users must put rpnreg.dll in their windows\system directory.

## **Registration**

RPN Calculator is freely distributable as ShareWare. If you find this program useful, please send \$20 to:

Tom Boldt  
2913 Castlebridge Drive  
Mississauga, Ontario, Canada  
L5M 5T2

Phone: (905) 814-9882

As an incentive to register RPN Calculator, some functions are disabled for unregistered users.

To report bugs or request additional features, email [tpboldt@hookup.net](mailto:tpboldt@hookup.net).

The latest version of RPN Calculator can be found at <http://www.hookup.net/~tpboldt>.



## **Requirements**

RPN Calculator runs in any of the following environments:

- Microsoft\* Windows\* 95.
- Microsoft\* Windows NT\* 4.0 (Intel\* only).

\* Microsoft, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation. Intel is a registered trademark of Intel Corporation.

## Legend

a	<u>real</u>
#a	<u>integer</u>
'a'	<u>variable</u>
[a]	<u>vector</u>
[[a]]	<u>matrix</u>
{a}	<u>list</u>
<<a>>	<u>program</u>
a_u1	<u>unit</u>
(a)	<u>complex</u>
obj <sub>1</sub>	objects of any type

See also: [Reading stack diagrams](#)

## Reading stack diagrams

The following notation is used to signify specific object types:

a	<u>real</u>
#a	<u>integer</u>
'a'	<u>variable</u>
[a]	<u>vector</u>
[[a]]	<u>matrix</u>
{a}	<u>list</u>
<<a>>	<u>program</u>
a_u1	<u>unit</u>
(a)	<u>complex</u>
obj <sub>i</sub>	objects of any type

The objects initially on the stack are shown on the left side of the arrow (->). The objects returned by the function are displayed on the right side of the arrow. For each side of the stack, the right-most object is the object on Level 1. The left-most object must therefore be pushed onto the stack first.

### Example:

```
^          Power
          a b      ->      c
```

This function takes two real numbers and returns one real number. To push 2 and 3 on the stack, type 2<enter>3<enter>. The stack will now contain 2: 2, 1: 3. In this case a will be 2 and b will be 3. Most functions require an <enter> to execute the function. Most of the one character functions do not require an <enter>. In this case just type the power symbol (^).

## Special Keys

The following key strokes perform a special action:

Ctrl+-	Inserts negative sign (-), as opposed to performing the subtract command
Ctrl+.	Inserts degree symbol (°)
Ctrl+A	Inserts Angstrom symbol (Å)
Ctrl+U	Inserts micron symbol (μ)
<Enter>	Duplicates an object when the edit field is empty
<Delete>	Drops an object when the edit field is empty
\	Swaps two objects when the edit field is empty

## Functions Reference

When you open the Tools menu, the status bar provides a brief explanation of the currently highlighted function.

To get more help for a specific function, highlight the function on the Tools menu and press F1. The help topic for the selected function displays.

On Windows NT 3.51, the menu option must be chosen with the keyboard, not the mouse, in order for F1 to work. To use the mouse, choose *Help | What's This?* and then select a menu option.

