Example

Zoom (125); changes the percentage magnification of the contents of the main spreadsheet to 125 percent.

Example

WindowSize (753, 437); resizes the main spreadsheet window to 753 pixels (approximately 8 1/6 inches) wide by 437 pixels (approximately 4 3/4 inches) tall.

Year

Year (*DNumber*)

The *Year* function returns the year, an integer from 1904 to 2083 (inclusive), corresponding to the date number *DNumber*.

Argument	Description
DNumber	Specifies a valid date number (see page 54 of this manual for more on Time functions).

Examples

Year (31920) gives 1991.

Year (Date (91, 11, 14)) gives 1991. The Date function here returns the date number corresponding to 11/14/1991. This number is then used as an argument in the Year function.

Year (Now ( )) gives the current year.

Zoom

Zoom (*Size*)

The *Zoom* command changes the percentage of magnification at which the contents of a spreadsheet window are displayed. *Zoom* is only applicable to the spreadsheet that is the main window. This command is equivalent to Zoom settings at the bottom right corner of a spreadsheet window.

Argument	Description
Size	Specifies the percentage value at which to resize the spreadsheet display. The maximum value is 200 and the minimum is 25.

	top left corner of the screen. Acceptable entries are: "Rel" or 1 "Abs" or 2 If you don't provide an argument for How, ParaSheet moves the window relative to its current location.

Windows cannot be moved off the visible screen.

Examples

WindowMove (92, 92, "Abs"); moves the main spreadsheet 92 pixels (1 inch) to the right and 92 pixels (1 inch) down from the top left corner of the screen.

WindowMove (–184, –184, 1); moves the main spreadsheet 184 pixels (2 inches) to the left and 184 pixels (2 inches) up, relative to its current location.

WindowSize

WindowSize (*NewWidth, NewHeight*)

The *WindowSize* command resizes a selected spreadsheet window in the workspace.

Argument	Description
NewWidth	A number specifying the amount (in pixels) by which to resize the window horizontally. (Default window width is 677 pixels.)
NewHeight	A number specifying the amount (in pixels) by which to resize the window vertically. (Default window height is 453 pixels.)

Windows cannot be sized beyond the boundaries of the visible screen.

	continues with the next instruction in the macro.
WhileClause	Specifies an instruction or group of instructions that WILMA completes if *Condition* is true. If the *WhileClause* contains multiple statements, they must be enclosed in braces ({ }).

Example

```
While ( Row < 100 )
    {
    Select ( Col, Row );
    Enter ( Col + Row );
    Row := Row + 1;
    }
```

While a variable named Row is less than 100, this macro selects the cell represented by the variables Col and Row and enters the sum of Col and Row in that cell.

See Also

For more information on the *While* statement, see page 32 of this manual.

WindowMove

WindowMove (*XDirection, YDirection* [, *How*])

The *WindowMove* command moves a selected spreadsheet window to a new location in the workspace.

Argument	Description
XDirection	A number specifying the amount (in pixels) to move a spreadsheet window horizontally in the workspace
YDirection	A number specifying the amount (in pixels) to move a spreadsheet window vertically in the workspace
How	Specifies whether to move the window relative to its current location or from the

value, backs up one cell, and moves right *Offset* cells from that cell. If *Value* is larger than all the numbers in the first column of *Column*, VLookup searches to the last row in the range *Column* and moves right from there. Since VLookup only searches the leftmost column for *Value* until it finds an equal or higher value, the values in the leftmost column should be in ascending order.

Examples

	A	B	C	D
1				
2	31	66	188	320
3	42	56	157	275
4	53	112	83	215
5	65	324	201	245
6	69	44	227	201
7	84	65	199	198
8	99	101	175	189

VLookup (65, A2:D8, 2) displays 201, the value 2 cells to the right of A5, which contains 65.

VLookup (200, A2:D8, 3) displays 189, the value 3 cells to the right of the index column, in the last row of the range *column* (200 is higher than any value in the index column).

While

While (*Condition*)

WhileClause

The *While* statement causes WILMA to repeat an action as long as a certain condition is true.

Argument	Description
Condition	Specifies a condition that, as long as it's true, controls how many times an associated *WhileClause* executes. When the condition is no longer true, WILMA skips over the associated *WhileClause* and

VIndex (60,A2:D8) gives 2, the count of cells searched in column A **before** a value higher than 60 was located.

VLookup

VLookup (*Value, Column, Offset*)

The *VLookup* command searches from top to bottom in the column furthest left in a range specified by *Column* until it finds the specified *Value*. *VLookup* then moves right in that row *Offset* cells and returns that location.

VLookup returns the cell reference (for example G15). ParaSheet can use this reference to display the value in the referenced cell, which is the usual result of *VLookup*. However, you can also use *VLookup* wherever a range reference is appropriate — for example, as an argument in the Sum function.

Argument	Description
Value	Specifies the text or numeric value to be searched for. Enter a text value if the leftmost column contains text; a numeric value if the leftmost column contains numeric values.
Column	A range specifying the index column (the leftmost column of a lookup table) — that is, the column in which VLookup should search for *Value*. If *Column* contains more than one column, only the leftmost column will be searched, from the first cell to the last row of *Column*.
Offset	A positive integer representing the number of cells to move to the right of the leftmost (index) column. When *Value* is text, VLookup searches the leftmost column for an exact match. For example, if *Value* is "Total", VLookup will locate only "Total", not "total" or "TOTAL". If VLookup doesn't find an exact match, it searches down to the **last** row in the range *Column* before moving right. When *Value* is numeric, VLookup searches until it finds either *Value* or a number larger than *Value*. If it finds *Value*, VLookup stops at that cell, and moves right *Offset* cells. Otherwise, it searches until it locates a larger

Column

A range specifying the index column (leftmost column of a lookup table) — that is, the column in which VIndex should search for *Value*. If the range *Column* contains more than one column, VIndex will search only the leftmost column of the range, beginning with the first cell, and ending with the cell that's in the last row of the range.

When *Value* is text, VIndex searches for an exact match. For example, if *Value* is "Total", VIndex will locate only "Total", not "total" or "TOTAL". If VIndex doesn't find an exact match, it gives the count of **all** the cells in the leftmost column of the range specified by *Column* (starting with 0), up to the last row in *Column*.

When *Value* is numeric, VIndex searches until it finds either *Value* or a number larger than *Value*. If it finds *Value*, VIndex gives the count of the cells searched (beginning with zero). Otherwise, it searches until it locates a larger value, backs up one cell, and gives the count of the cells searched up to that cell. If *Value* is larger than all the numbers in the first column of the range *Column*, VIndex gives the count of all the cells in that column (starting with 0), up to the last row in *Column*. Since VIndex only searches the leftmost column for *Value* until it finds an equal or higher value, the values in the leftmost column should be in ascending order.

Examples

	A	B	C	D
1				
2	31	66	188	320
3	42	56	157	275
4	53	112	83	215
5	65	324	201	245
6	69	44	227	201
7	84	65	199	198
8	99	101	175	189

VIndex (69,A2:D8) gives 4, the count of cells searched in column A to locate the value 69.

Examples

	A	B
1	35	
2	25	
3	29	
4	42	
5	27	
6	39	
7	40	
8	30	
9	21	
10	42	

Variance (A1:A10) gives *57.777778*, the sample variance of the values in cells A1:A10.

The sample variance, Variance, is an estimate of the population variance, based on a sample of the population. The population variance is calculated with *n* in the denominator of the variance formula, whereas the sample variance is calculated with *n* –1 in the denominator (*n* is the number of observations in your sample). To calculate the population variance based on the sample above, use the following formula:

Variance (*A1:A10*) * (*Count* (*A1:A10*) –*1*) / *Count* (*A1:A10*)
(*Count* (*A1:A10*) is the number of observations (*n*) in the sample above.)

VIndex

VIndex (*Value, Column*)

The *VIndex* command searches from top to bottom in the column furthest left in the range specified by *Column* until it finds the specified *Value*, and returns the count of the cells searched. As it searches, *VIndex* counts cells starting at zero. For example, if two cells are searched to locate *Value*, *VIndex* returns 1.

Argument	Description
Value	Specifies the text or numeric value to be searched for. Enter a text value if the leftmost column contains text; a numeric value if the leftmost column contains numeric values.

Examples

True ( ) = True ( ) gives a value of True (1).

IfX (True ( ), "Sales are up", "Sales are down") always gives "Sales are up" because the condition evaluates to True.

Choose (0, True ( ), False ( )) gives True (1).

Variance

Variance (*List*)

The *Variance* function calculates the sample variance of the values in a specified list, using the following formula:

$\sum (X - Avg)^2 / (n - 1)$

where each X is a value in the sample, Avg is the average of the values in the sample, and n is the number of values in the sample.

Argument	Description
List	Specifies one or more numeric values, for which ParaSheet calculates the variance. Multiple list items must be separated by commas. Acceptable entries are: A numeric value or values (formulas or constants) Cells or ranges containing numeric values Any combination of the above Text and null values are ignored. If you provide only a single value for *List*, ParaSheet returns a !DIV error.

20	Month YYYY	August 1992
21	Mon YYYY	Aug 1992
22	YY.M.D	92.8.2
23	YYYYMMDD	19920802

Format number	Time format	Example
0	No time	No time
1	H:MM AM/PM	4:36 PM
2	HH:MM AM/PM	04:36 PM
3	H:MM:SS AM/PM	4:36:44 PM
4	HH:MM:SS AM/PM	04:36:44 PM
5	HH:MM (24 Hr)	16:36
6	HH:MM:SS (24 Hr)	16:36:44
7	HHMM	0436

Examples

Time2C (32156.5, 1, 0) gives the string "1/15/92".

Time2C (32156.516875, 12, 1) gives the string "1-15-92 12:24 PM".

Time2C (32156.5, 17, 3, 1) gives the string "12:00:00 PM 15 Jan 1992".

Time2C (Now ( ), 17, 3, 1) gives the *current* time and date in the format shown in the previous example. If ParaSheet evaluated this function at exactly 12 PM on 1/15/92, it would give "12:00:00 PM 15 Jan 1992".

True (Logical True)

True ( )

The *True* function returns the logical value *True*, which is equivalent to the numeric value 1.

There are no arguments for *True*, but parentheses must be included.

TFormat, where 0 means that no time is included in the text string. See Table 2-3 below for these values and their corresponding formats.

TimeFirst — Specifies either 0 or a nonzero value, where 0 means that the date in *DFormat* precedes the time in *TFormat* in the resulting text string, and a nonzero value means that the time in *TFormat* precedes the date in *DFormat* in the string. If *TimeFirst* is not specified, ParaSheet assumes zero, so that the date precedes the time.

Format number	Date format	Example
0	No date	No date
1	M/D/YY	8/2/92
2	M/D/YYYY	8/2/1992
3	M/D	8/2
4	M/YY	8/92
5	D-Mon-YY	2-Aug-92
6	DD-Mon-YY	02-Aug-92
7	D-Mon-YYYY	2-Aug-1992
8	DD-Mon-YYYY	02-Aug-1992
9	D-Mon	2-Aug
10	DD-Mon	02-Aug
11	Mon-YY	Aug-92
12	M-D-YY	8-2-92
13	M-D-YYYY	8-2-1992
14	Month D, YYYY	August 2, 1992
15	Mon D, YYYY	Aug 2, 1992
16	D Mon YYYY	2 Aug 1992
17	DD Mon YYYY	02 Aug 1992
18	Month D	August 2
19	Mon D	Aug 2

Time

Time (*Hour, Minute, Second*)

The *Time* function returns the date number of the time specified by an hour, minute, and second. The returned date number is a decimal fraction from 0 (for 00:00:00 or 12:00:00 midnight) to .99999 (for 23:59:59 or 11:59:59 PM).

Argument	Description
Hour	An integer from 0 to 23, inclusive, specifying the hour
Minute	An integer from 0 to 59, inclusive, specifying the minute
Second	An integer from 0 to 59, inclusive, specifying the second

Example

Time (16, 14, 30) gives the date number 0.676736.

Time2C (Date & Time to Text String)

Time2C (*DNumber, DFormat, TFormat* [, *TimeFirst*])

The *Time2C* function returns the text string representing the date and time specified by a date number, using the formats specified by the date format number (*DFormat*) and the time format number (*TFormat*). Either the time or date can appear first, depending on the value of *TimeFirst*.

Argument	Description
DNumber	Specifies a valid date number (see page 54 of this manual for more on Time functions).
DFormat	Specifies an integer representing the format of the date part of the text string to be returned. The values 0 to 23 are acceptable for *DFormat*, where 0 means that no date is included in the text string. See Table 2-2 below for these values and their corresponding formats.
TFormat	Specifies an integer representing the format of the time part of the resulting text string. The values 0 to 7 are acceptable for

EndChar	An positive integer specifying the character at which to stop selecting text.
BeginWord	A positive integer specifying the word from which *BeginChar* should start selecting text. If you don't specify *BeginWord*, ParaSheet counts characters from the beginning of text in the specified object.
EndWord	A positive integer specifying the word from which *EndChar* should count to find the end of the selection. If you don't specify *EndWord*, ParaSheet counts characters from the beginning of text in the specified object.
ObjectID	Specifies the name or number of the chart text object in which text will be selected. If you don't provide an *ObjectID*, ParaSheet selects text in the currently selected chart text object.

When selecting text in a chart text object, you must first use ChartSelect to select the chart text object.

To place the insertion point at the beginning of text in a chart text object, specify 1 for the Where argument. To place the insertion point at the end, specify a number greater than the number of characters in the chart text object.

Examples

TextSelect (9); positions the insertion point at the ninth character in the currently selected chart text object.

TextSelect (1, 6); selects the first through sixth characters starting at the beginning of the currently selected chart text object.

TextSelect (2, 2); selects the second character in the currently selected chart text object).

TextSelect (3, 4, 2, 7); selects text starting at the third character of the second word through the fourth character of the seventh word.

TextSelect (0); positions the insertion point before the first character in the currently selected chart text object.

TextSelect (33333); positions the insertion point after the last character in the currently selected chart text object, assuming it contains a maximum of 33333 characters.

Tan (Tangent)

Tan (*Angle*)

The *Tan* function returns the tangent of a specified angle given in radians. To convert a degree angle to radians, use the D2R function or multiply it by pi/180, as shown in the example below.

Argument	Description
Angle	Specifies the size of an angle in radians. Acceptable entries are: A numeric value (formula or constant) excluding –∞ and +∞ A cell that contains a numeric value A range, the first cell of which contains a numeric value

Examples

Tan (45) gives 1.619775.

Tan (D2R (45)) gives 1.

Tan (45 * Pi ( ) / 180) gives 1.

TextSelect

TextSelect ({ *Where* | *BeginChar, EndChar* [, *BeginWord, EndWord* [, *ObjectID*]] })

The *TextSelect* command either specifies where to position the insertion point, or provides a starting and ending point for a text selection in a chart text object.

Argument	Description
Where	A positive integer specifying where to position the insertion point. If you specify *Where,* you can't provide other arguments.
BeginChar	An positive integer specifying the character at which to start selecting text. ParaSheet counts characters from the beginning of the chart text object starting with the integer 1.

Sum

Sum (*List*)

The *Sum* function adds up the numeric values in a specified list.

Argument	Description
List	Specifies a list of one or more values to be added together. Multiple list items must be separated by commas. Acceptable entries are: A numeric value or values (including formulas or constants) A text value or values A null value or values A cell or range containing numeric, text, or null values, or any combination of these Sum ignores non-numeric values in *List*.

Examples

	A	B	C
1	Session 1	Session 2	Session 3
2			
3	9	2	5
4	41	31	28
5	12	48	25
6	16	51	20
7	12	23	10

Sum (A3:A7) gives 90.

Sum (A1:A7) gives 90. Cells A1 and A2 are ignored in this calculation, because they don't contain numeric values.

Sum (A3:C3) gives 16.

Sum (B3, B5, 21 / 4, 16, C3:C7) gives 159.25.

Sum (A3:C7) gives 333.

Argument	Description
List	Specifies the list of values for which ParaSheet finds the standard deviation. Acceptable entries are: Numeric values (formulas or constants) Cells or ranges containing numeric values A combination of these

Example

	A	B
1	35	
2	25	
3	29	
4	42	
5	27	
6	39	
7	40	
8	30	
9	21	
10	42	

StdDev (A1:A10) gives 7.60117, the sample standard deviation of the values in cells A1:A10.

The sample standard deviation, StdDev, is an estimate of the population standard deviation, based on a sample of the population. The population standard deviation is calculated with *n* in the denominator of the variance formula, whereas the sample standard deviation is calculated with *n* –1 in the denominator (*n* is the number of observations in your sample). See the description of the Variance function. To calculate the population standard deviation based on the sample above, use the following formula:

StdDev (A1:A10) * *Sqrt* ((*Count* (A1:A10) –1) / *Count* (A1:A10))

(Count (A1:A10) is the number of observations (*n*) in the sample above.)

Example

SOYD (5000, 200, 5, 2) returns $1280.00 (in a cell formatted for Currency). This is the amount to depreciate a $5000 investment the second year calculated by a sum of the years' digits depreciation. The investment is assumed to have a residual (salvage) value of $200 after five years.

Sqrt (Square Root)

Sqrt (*Number*)

The *Sqrt* function returns the positive square root of a specified number.

Argument	Description
Number	Specifies the number for which the square root is calculated. Acceptable entries are: A non-negative numeric value (formula or constant) A cell that contains a non-negative numeric value A range, the first cell of which contains a non-negative numeric value

Examples

Sqrt (25) gives 5.

Sqrt (20 + 5) gives 5.

Sqrt (D2) gives 3 when cell D2 contains the value 9.

StdDev (Standard Deviation)

StdDev (*List*)

The *StdDev* function returns the sample standard deviation (using the method of least squares) of the numeric values in a specified list by taking the square root of the variance (difference). The formula used is:

Sqrt (*Variance* (*List*))

Example

```
SortKeyReset ( "Cols", A1:D14 );
Sort ( "Yes", A1:D14 );
SortKeySet ( A7:D7, "Descend", 1 );
Sort ( "Yes", A1:D14 );
```

Sorts records in the range A1:D14 (ignoring case) according to the current settings for priority and order, resets the record in A7:D7 to first priority and in descending order, and re-sorts the records in A1:D14.

See Also

For a comprehensive example, see the *Sort* command.

SOYD (Sum of the Years' Digits)

SOYD (*Inv, Residual, Term, Period*)

The *SOYD* function returns the depreciation amount for an investment for a given period, using a sum of the years' digits depreciation. This function uses the original investment, the number of years over which to depreciate, the residual value at the end of the term, and the period for which the depreciation amount is to be calculated. ParaSheet uses the following formula for *SOYD*:

[2 * (1 + ***Term*** – ***Period***) * (***Inv – Residual***)] / [***Term*** * (***Term*** + 1)]

Argument	Description
Inv	A number specifying the initial investment
Residual	A number specifying the remaining value at the end of the term
Term	A number specifying the total length of depreciation time
Period	A number specifying the period for which to calculate the depreciation amount

Range	Specifies a sorted row, a column (either of these can be a named range), or range of records.

Example

SortKeyReset ("Rows", A1:D14);

Sort ("Yes", A1:D14);

Ensures sort keys are set to the ParaSheet default for records arranged in rows for the sort range A1:D14, then sorts records in that range (ignoring case).

See Also

For a comprehensive example, see the *Sort* command.

SortKeySet

SortKeySet (*Record, Direction* [, *Priority*])

The *SortKeySet* command determines the sort order for a record. It also sets its sort priority relative to other records in a sort range. *SortKeySet* is dependent on the *SortKeyReset* command to first set a sort range. This command is equivalent to assigning sort direction and priority to a cell range using the Set Priority buttons and sort key settings in the Sort Panel (*Sort* in the Edit menu).

Argument	Description
Record	Specifies cells in a column or row to be sorted. Acceptable entries are: Cell references A named range
Direction	Specifies the sort direction. Acceptable entries for sort direction of a record are: "Ascend" or 1 "Descend" or 2
Priority	A positive integer specifying the order in which the record is sorted. This integer must not exceed the number of records being sorted.

command is equivalent to setting a cell range to the Ignore key in the Sort panel (*Sort* in the Edit menu).

Argument	Description
Record	Specifies cells in a record to be ignored in a sort. Acceptable entries are: Cell references A named range

Example

```
SortKeyReset ( "Cols", A1:D14 );
Sort ( "Yes", A1:D14 );
SortKeyDel ( A2:D2 );
Sort ( "Yes", A1:D14 );
```

Sorts records in the range A1:D14 (ignoring case) according to the current settings for priority and order, resets the sorting direction for the record in A2:D2 to ignore, then re-sorts the records in A1:D14.

See Also

For a comprehensive example, see the *Sort* command.

SortKeyReset

SortKeyReset (*RecordsIn* , *Range*)

The *SortKeyReset* command determines whether records to be sorted are arranged in rows or columns. It also sets all record key settings for a range to Ignore except the first, which is set to ascending order. This command is equivalent to default settings in the Sort panel (*Sort* in the Edit menu).

Argument	Description
RecordsIn	Specifies whether to sort by rows (data is arranged in columns) or by columns (data is arranged in rows). Acceptable entries are: "Rows" or 1 "Cols" or 2

Sort will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

Sort ( ); brings up the Sort panel.

Sort ("No", A1:D14); specifies a case-sensitive sort of the contents of the range A1 through D14. Any other sort settings (such as priority and arrangement of records in columns or rows) are not affected.

SortKeyReset ("Cols", A1:D14);

SortKeyDel (A1:D1);

SortKeySet (A4:D4, "Ascend");

SortKeySet (A2:D2, "Descend");

SortKeySet (A7:D7, "Ascend", 1);

Sort ("Yes", A1:D14);

The SortKeyReset instruction specifies records are arranged in columns, and resets the sorting key to the ParaSheet default (the first record is the only key, and is sorted in ascending order) for the range A1 through D14. The SortKeyDel instruction resets the default sorting key (A1:D1) to Ignore. The SortKeySet instructions give the record in A7:D7 to first priority and set it to sort in ascending order. The record in A2:D2 has second priority and sorts in descending order.

The record in A4:D4 has third priority and sorts in ascending order. Note that even though A2:D2 follows A4:D4 in the sequence of instructions it is sorted first — the SortKeySet command has no effect on priority unless you provide a priority argument. The Sort command instructs ParaSheet to execute the actual sort.

For more information on the sorting process, see Chapter 10 in the *Tutorial and Documentation* manual.

SortKeyDel

SortKeyDel (*Record*)

The *SortKeyDel* command resets the sorting direction for a record to Ignore. It does not change the sorting priority of records relative to one another. *SortKeyDel* is dependent on SortKeyReset to first set a sort range. This

Argument	Description
Inv	A number specifying the initial investment
Residual	A number specifying the remaining value of the investment at the end of the depreciation term
NumPeriods	A number specifying the number of time periods over which to depreciate

Example

SL (6000, 500, 5) gives $1100.00, the amount to depreciate a $6000 investment per year calculated by the straight-line depreciation method. The residual or salvage value of the investment is assumed to be $500 after a depreciation term of five years.

Sort

Sort ([*Options* [, *Selection*]])

The *Sort* command arranges the contents of a range of records according to ParaSheet sorting order. If you provide arguments, Sort is dependent on SortKeyReset to first set a sort range. If you don't provide any arguments, the Sort panel appears. This command is equivalent to selecting a sort range in a spreadsheet and setting the *Ignore Case* option in the Sort panel (*Sort* in the Edit menu).

Argument	Description
Options	Specifies whether uppercase letters are sorted among lowercase letters (case is ignored), or sorted after all lowercase letters are sorted. Acceptable entries are: "Yes" or 1 (to ignore case) "No" or 0 (for a case-sensitive sort)
Selection	Specifies a row, a column (either of these can be a named range), or range of records. If you don't provide a selection, ParaSheet sorts the current selection in ascending order.

SizeSet

SizeSet (*What, Size* [, *Selection*])

The *SizeSet* command sets the height of a row or the width of a column (in pixels).

Argument	Description
What	Specifies whether rows or columns are resized. Acceptable entries are: "Row" or 1 "Col" or 2 "Both" or 3
Size	A number specifying the row height or column width, in pixels. (Default row height is 16 pixels and default column width is 78 pixels.)
Selection	Specifies a range of rows or columns. If you don't provide a selection, ParaSheet resizes the rows or columns in the current selection.

Examples

SizeSet ("Col", 156, F:F); sets column F to 156 pixels (two times its usual default width).

SizeSet (1, 48, MyNamedRange); sets all the rows in the named range MyNamedRange to 48 pixels (three times their usual default height).

SL (Straight-Line Depreciation)

SL (*Inv, Residual, NumPeriods*)

The *SL* function calculates the straight-line depreciation amount per period of an initial investment, given the number of time periods over which to depreciate, and the residual value at the end of the term. The formula used is:

(*Inv* – *Residual*) / *NumPeriods*

Examples

Sign (C6) returns –1 when cell C6 contains –.14324.

Sign (0) returns 0.

Sign (5) returns 1.

Sign (–5) returns –1.

Sin (Sine)

Sin (*Angle*)

The *Sin* function returns the sine of an angle given in radians. Its value is always between –1 and 1, inclusive. To convert a degree value to radians, use the D2R function or multiply it by pi/180, as shown in the example below.

Argument	Description
Angle	Specifies the size of an angle in radians. Acceptable entries are: A numeric value (formula or constant) excluding –∞ and +∞ A cell that contains a numeric value A range, the first cell of which contains a numeric value To convert a degree value to radians, multiply it by pi/180, as shown in the example below.

Examples

Sin (Pi ( ) / 2) gives 1.

Sin (D2R (30)) gives 0.5, the sine of 30 degrees.

Sin (30 * Pi ( ) / 180) gives 0.5, the sine of 30 degrees.

	default number of rows in a spreadsheet is 2048.)
Size2	When resetting both rows and columns, a positive integer specifying the number of columns. (The default number of columns in a spreadsheet is 256.)

Examples

SheetSizeSet ("Rows", 100); sets the number of rows in the main spreadsheet to 100.

SheetSizeSet (2, 50); sets the number of columns in the main spreadsheet to 50.

SheetSizeSet ("Both", 200, 100); sets the number of rows to 200 and the number of columns to 100 in the main spreadsheet.

Sign

Sign (*Number*)

The *Sign* function returns whether a specified number is positive, zero, or negative.

Returns	Description
1	Indicates the number is positive
0	Indicates the number is zero
–1	Indicates the number is negative

Argument	Description
Number	Specifies the number to be checked for its sign. Acceptable entries are: A numeric value (including a formula or constant) A cell containing a numeric value A range, the first cell of which contains a numeric value

There are no arguments for *SelectAll*, but parentheses must be included.

SendToBack

SendToBack ([*ObjectList*])

The *SendToBack* command places specified objects behind other objects. This command is equivalent to the *SendToBack* command in the Layout submenu.

Argument	Description
ObjectList	One or more object names or numbers; or a combination of names and numbers. Object names must be surround- ed by quotation marks. If you don't provide an object list, ParaSheet sends the currently selected objects to the back.

Example

SendToBack (3, "OilConsumption"); causes object number 3 and the chart named OilConsumption to appear behind all other objects.

SheetSizeSet

SheetSizeSet ([*What, Size1* [, *Size2*]])

The *SheetSizeSet* command changes the number of columns and rows in a spreadsheet. If you don't provide any arguments, the Preferences panel appears. This command is equivalent to settings in the Spreadsheet Size area of the Preferences panel (the *Preferences* command in the Info menu).

Argument	Description
What	Specifies the spreadsheet range to be changed. Acceptable entries are: "Rows" or 1 "Cols" or 2 "Both" or 3
Size1	A positive integer specifying the number of rows or columns for the spreadsheet. When resetting both rows and columns, a positive integer specifying the number of rows. (The

Examples

Select (3); selects object number 3.

Select (3,4); selects cell C4.

Select (WaterLevels); selects a named range called WaterLevels.

Select ("/me/Data/Spreadsheets/Resources"); selects a spreadsheet named Resources. Resources is now the main spreadsheet window, so subsequent actions will affect it.

SelectAdd

SelectAdd (*List*)

The *SelectAdd* command selects multiple charts, ranges, or cells.

Argument	Description
List	Specifies one or more object names or numbers, or ranges or cells for a multiple selection.

Examples

SelectAdd (1, 2, 3, 4, 5);

ObjMove (92, 184);

Selects objects numbered 1 through 5, and moves them 1 inch to the right and 2 inches down.

SelectAdd (Range1, Range2);

Move (Range1, Range2);

Selects two named ranges called Range1 and Range2 and moves the contents of Range1 to Range2.

SelectAll

SelectAll ( )

The *SelectAll* command selects either all spreadsheet cells, all objects in the spreadsheet, or the contents of a chart text object, depending on what is currently selected. This command is equivalent to the *Select All* command in the Edit menu.

Second (Second of the Minute)

Second (*DNumber*)

The *Second* function returns an integer from 0 to 59 (inclusive), corresponding to the second of the minute for the date number you specify. 0 is on the minute, 1 is one second after the minute, and 59 is one second before the next minute.

Argument	Description
DNumber	Specifies a valid date number (see page 54 of this manual for more on Time functions).

Examples

Second (33203.516875) gives 18.

Second (Time (12, 30, 0)) gives 0. The Time function here gives the date number corresponding to 12:30:00 PM, which is then used as an argument in the Second function.

Select

Select (*What*)

The Select command selects the contents of a cell, a range of cells, a chart object, or a spreadsheet window.

Argument	Description
What	Specifies a cell, coordinate reference, range of cells, named range, object name or number, or filename. When selecting a spreadsheet window, if you don't provide a path name, ParaSheet assumes the spreadsheet is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename and path name must be surrounded by quotation marks. Object names must also be surrounded by quotation marks.

SaveTo will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

SaveTo ("/me/Data/LotusFiles/AustralianParrots123", "Lotus"); saves the active spreadsheet under the name AustralianParrots123.wk1 in Lotus 1-2-3 format in the specified folder.

SaveTo ("ResourcesText", 2); saves the active spreadsheet under the name ResourcesText in UNIX text format.

Scroll

Scroll (*Type, Direction* [, *Amount*])

The *Scroll* command scrolls the spreadsheet display horizontally or vertically.

Argument	Description
Type	Specifies the type of increment by which to scroll the spreadsheet. Acceptable entries are: "Page" or 1 "Pixel" or 2
Direction	Specifies the scrolling direction. Acceptable entries are: "Horiz" or 1 "Vert" or 2
Amount	When scrolling by page or by pixel, an integer specifying the amount by which to scroll. If you don't provide an *Amount*, ParaSheet uses the default value 1.

Examples

Scroll ("Page", "Horiz", 3); scrolls the current spreadsheet three pages to the right.

Scroll ("Pixel", "Vert", –160); scrolls the current spreadsheet up 10 rows.

SaveAs will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

SaveAs ("/me/Data/Spreadsheets/Birds/AustralianParrots"); saves the active spreadsheet (assume its name is Parrots) under the name AustralianParrots in the specified folder.

SaveAs ("Territory"); saves the active spreadsheet (assume its name is Habitat) under the name Territory in the current folder.

SaveTo

SaveTo ([*Filename, FileType*])

The *SaveTo* command prepares the active spreadsheet for use in another application or on another computer. If you don't provide any arguments, the Save To panel appears. This command is equivalent to choices in the Save To panel (the *Save To* command in the Document menu).

Argument	Description
Filename	Specifies the filename and path name (if necessary) to which the active spreadsheet will be exported. If you don't provide a path name, ParaSheet saves *Filename* to the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename and path name must be surrounded by quotation marks. If you don't provide a filename, the Save To panel appears.
FileType	Specifies the file format to which the spreadsheet is exported. *FileType* must be surrounded by quotation marks. Acceptable entries are: "Lotus" or 1 "UNIXText" or 2 "MacText" or 3 "PCText" or 4

	other than the current working folder, ParaSheet changes the working folder to the new path. The filename and path name must be surrounded by quotation marks. If you don't provide a filename and the spreadsheet is already named, ParaSheet saves the active spreadsheet to the filename and path to which it was last saved. Otherwise, the Save panel appears

Save will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

Save ("Resources"); saves the active spreadsheet under the name Resources in the current folder.

Save ( ); brings up the Save panel when the active spreadsheet is unnamed, or saves the active spreadsheet if it already has a name.

Save ("/me/Data/Spreadsheets/TropicalBirds"); saves the active spreadsheet under the name TropicalBirds in the specified path.

SaveAs

SaveAs ([*Filename*])

The *SaveAs* command stores the active spreadsheet under another name and/or in another location. This command is equivalent to the *Save As* command in the Document menu.

Argument	Description
Filename	Specifies the filename and path name (if necessary) to which the active spreadsheet will be saved. If you don't provide a path name, ParaSheet saves *Filename* in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename and path name must be surrounded by quotation marks. If you don't provide a filename, the Save panel appears.

NumDigits is more than the number of digits to the right of the decimal point, the result of Round will be the specified number.

If *NumDigits* is negative, ParaSheet takes the absolute value of *NumDigits* and uses it to round *Number* to the nearest power of ten. When the absolute value of a negative *NumDigits* is greater than or equal to the number of digits to the left of the decimal point, ParaSheet returns 0 — unless *Number* multiplied by 2 is greater than ten to the power of *NumDigits*, in which case it returns the nearest power of ten.

Examples

Round (252.156, 3) gives 252.156.

Round (252.156, 2) gives 252.16.

Round (252.156, 1) gives 252.2.

Round (252.156, 0) gives 252.

Round (252.156, –1) gives 250.

Round (252.156, –2) gives 300.

Round (252.156, –3) gives 0.

Round (654.156, –3) gives 1000.

Round (654.156, –4) gives 0.

Save

Save ([*Filename*])

The *Save* command stores changes made to the active spreadsheet since that spreadsheet was last saved. This command is equivalent to the *Save* command in the Document menu.

Argument	Description
Filename	Specifies the filename and path name (if necessary) to which the spreadsheet will be saved. If you don't provide a path name, ParaSheet saves *Filename* in the current working folder. If you provide a path name

	A non-negative integer
	A formula that evaluates to a non-negative integer
	A cell or range that contains a non-negative integer value
	When the value of *NumChars* exceeds the total number of characters in the string, Right returns all the characters in the string.

Examples

Right ("J. & J. Waterson and Co.", 16) gives "Waterson and Co.", the 16 characters furthest right in the string.

Right (H16, J20) gives "Scheduling Tips" when cell H16 contains the string "Scheduling Tips" and cell J20 contains 35.

Round (Rounded Value)

Round (*Number, NumDigits*)

The *Round* function returns the value of a specified number, rounded to the specified number of decimal places.

Argument	Description
Number	Specifies the number to be rounded.
	Acceptable entries are:
	A numeric value (formula or constant)
	A cell that contains a numeric value
	A range, the first cell of which contains a numeric value
NumDigits	An integer specifying the number of decimal places to which the number is rounded
	The following rules apply:
	If *NumDigits* is zero, then *Number* is rounded to the nearest integer.
	If *NumDigits* is positive, then *Number* is rounded to the specified number of decimal places to the right of the decimal point. If

Argument	Description
Setting	Specifies whether to request confirmation before reverting to the last saved version of a spreadsheet. Acceptable entries are: "Yes" or 1 to revert without confirmation "No" or 0 to request confirmation If you don't provide a setting, the Revert panel appears.

RevertToSaved will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

RevertToSaved ("Yes"); closes the main spreadsheet without storing any unsaved changes and opens the last saved version.

RevertToSaved ( ); brings up the Revert panel.

Right (Right Substring)

Right (*String, NumChars*)

The *Right* function returns a specified number of characters (including spaces) furthest right in a text string.

Argument	Description
String	Specifies the text string. Acceptable entries are: A sequence of ASCII characters surrounded by quotation marks A cell that contains a text string A range, the first cell of which contains a text string A formula that evaluates to a text string
NumChars	Specifies the number of characters to be returned. Acceptable entries are:

Example

```
\\ Macroname: 1991Budget
...
If ( Result <= Budget )
    {
    Value := Consolidation ( );
    Alert ( Value );
    }
...
\\ Macroname: Consolidation
...
If ( Result = 1 )
Return ( "Consolidation finished without error." );
Else
Return ( "Consolidation aborted." );
```

In the first macro, the results of an action are assigned to variables named Result and Budget. If the value of Result is less than or equal to the value of Budget, this macro calls another macro named Consolidation. The second macro returns a text string to the first macro indicating the result of the consolidation. This text string is then assigned to the local variable Value, which in turn is used as an argument for the Alert command.

For more information on the Return statement, see the "Returning Results" section on page 37 of this manual.

RevertToSaved

RevertToSaved ([*Setting*])

The *RevertToSaved* command loads the last saved version of a spreadsheet, erasing the most recent unsaved changes. This command is equivalent to the *Revert to Saved* command in the Document menu.

	The ParaSheet default is for automatic redrawing.

Example

```
Redraw ( "No" );
Select ( B3:B8 );
ChartNew ( "Pie" );
PieStartAngleSet ( 30 );
ObjInfo ( "LandfillContents" );
ObjMove ( 92, 184, C1 );
Redraw ( "Yes" );
```

Optimizes redrawing, creates a pie chart using data in cells B3 through B8, resets the starting angle at 30 degrees, renames the chart to LandfillContents, and moves it to a new location. Then, this macro enables automatic redraw. Temporarily disabling automatic redraw makes this macro execute faster, because complete redrawing doesn't occur automatically until the Redraw command is used (or until the macro terminates).

Return

Return ([*Value*])

The *Return* statement stops a running macro. If the macro was called by another macro or function, the Return statement can optionally pass a value back to the calling macro or function.

Returns	Description
A value	A number, text, or logical value, depending on the specified value.
1	If you don't provide an argument for Value.

Argument	Description
Value	Specifies a text, logical, or numeric value, or combination of values that produce a single value when calculated.

CalculateBy	Specifies the order in which formulas and macros recalculate. Acceptable entries are: "Natural" or 1 to recalculate in order of dependency "ByRow" or 2 to recalculate sequentially by row "ByCol" or 3 to recalculate sequentially by column If you don't provide an argument for *CalculateBy*, recalculation order stays the same.

RecalcOptions will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

RecalcOptions ("Auto", "Natural",); causes ParaSheet to automatically recalculate spreadsheets only when necessary.

RecalcOptions (2, 2); causes ParaSheet to recalculate the main spreadsheet by rows only when you use the Recalc or RecalcAll command (or choose Recalc or Recalc All from the Sheet menu).

Redraw

Redraw (*Setting*)

The *Redraw* command reduces the amount of spreadsheet redrawing that takes place after every macro action. This command is particularly useful to save redrawing time when working with large, graphically complex spreadsheets.

Argument	Description
Setting	Specifies whether automatic redraw is enabled. Acceptable entries are: "Yes" or 1 for automatic redrawing "No" or 0 to optimize redrawing

Examples

Recalc ("Resources"); recalculates all spreadsheet formulas in the spreadsheet named Resources.

Recalc ( ); recalculates the current spreadsheet.

RecalcAll

RecalcAll ( )

The *RecalcAll* command recalculates all open spreadsheets. It also recalculates links between all open spreadsheets and any supporting spreadsheets directly linked to them. ParaSheet does not recalculate linked spreadsheets beyond the first level of links; that is, supporting spreadsheets not directly linked to open spreadsheets. (For more information on linked spreadsheets, refer to the "Linking Spreadsheets" section on page 89 of the *Tutorial and Documentation* manual.) This command is equivalent to the *Recalc All* command in the Sheet menu.

RecalcAll will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

There are no arguments for *RecalcAll*, but parentheses must be included.

RecalcOptions

RecalcOptions (*How* [, *CalculateBy*])

The *RecalcOptions* command sets the method and order of recalculation for the spreadsheet. Recalculation options correspond to choices in the Recalculation Options panel. This command is equivalent to settings in the Recalculation Options panel (the Recalc Options command in the Sheet menu).

Argument	Description
How	Specifies whether recalculation is automatic. Acceptable entries are: "Auto" or 1 "Manual" or 2

Examples

Random ( ) can give 0.9876546. (The results of Random vary.)

Random (10) gives a random integer between 0 and 10, excluding 10.

Random ( ) * 10 gives a random rational number between 0.0 and 10.0, excluding 10.0.

Int (Random ( ) * 10) gives a random integer between 0 and 10 excluding 10. The formula Random()*10 here gives rational numbers between 0 and 10, excluding 10. This value is then used as an argument for the Int function, which gives the integer portion of the value. This example is equivalent to Random (10).

Random (–15) gives a random integer between 0 and –15, excluding –15.

Recalc

Recalc ([*Filename*])

The *Recalc* command activates the specified spreadsheet and recalculates that spreadsheet according to the current recalculation options. This command is equivalent to the Recalc command in the Sheet menu.

Argument	Description
Filename	Specifies the filename and path name (if necessary) of the spreadsheet to be recalculated. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename and path name must be surrounded by quotation marks. If you don't provide a filename, ParaSheet recalculates the current spreadsheet.

Recalc will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

	Acceptable entries are:
	A number
	A formula
	A cell reference
	A named range

Example

R2D (Atan (1)) gives 45 degrees.

Random (Random Value)

Random ([*Maximum*])

The *Random* function returns a random rational number between 0 and 1, including 0 but not 1. If you specify an argument, Random returns a random number between 0 and *Maximum*, including 0 but not including *Maximum*. The results of the *Random* function are different with each use, and change each time ParaSheet completes a recalculation.

Argument	Description
Maximum	Specifies the maximum number Random will return (a limit that Random will not exceed).
	Acceptable entries are:
	A numeric value (formula or constant)
	A cell that contains a numeric value
	A range, the first cell of which contains a numeric value
	If you don't specify *Maximum,* ParaSheet returns a number greater than or equal to 0 and less than 1. If you specify 0, ParaSheet returns 0.

PV (Present Value)

PV (*Pmt, Rate, NumPmts*)

The *PV* function calculates the present value of a series of payments (each of a specified amount) that collect a specific periodic interest rate over the specified number of payment periods. *PV* assumes payments are made at the end of each period. If payments are made at the beginning of each period, multiply the result by (1 + *Rate*). *PV* uses the formula:

[*Pmt* * (1 – ((1 + (*Rate*))$^{-NumPmts}$))] / *Rate*

Argument	Description
Pmt	A number specifying the amount of each payment
Rate	A number specifying the periodic interest rate
NumPmts	A number specifying the number of payment periods

Examples

PV (50, 15% / 12, 12) gives $553.97.

PV (A5, B3 / 12, D2) gives a value of $7885.63 when cell A5 contains 200 (amount of each payment), cell B3 contains .1 (the annual interest rate), and cell D2 contains 48 (the number of payment periods).

R2D (Radians to Degrees)

R2D (*Number*)

The R2D function converts a number from radians to degrees, according to the following formula:

Number * 180 / π

R2D is particularly useful with mathematical functions, such as Acos, Asin, Atan, Atan2, Cos, Cot, Sin, and Tan.

Argument	Description
Number	Specifies the value to be converted from radians to degrees.

	specify two double quotes (“”) with no intervening characters.
Message	Specifies a text message to appear in the panel. The text message must be less than 255 characters long, surrounded by quotation marks. If you don’t want a message, you must specify two double quotes (“”) with no intervening characters.
Button1	Specifies the text that appears in the default button. The number of characters in the text string must be less than or equal to 15. If you don’t want text on the button, you must specify two double quotes (“”) with no intervening characters.
Button2	Specifies the text that appears in the other button. The number of characters in the text string must be less than or equal to 15. If you don’t want text on the button, you must specify two double quotes (“”) with no intervening characters.
Button3	Specifies the text that appears in the third button. The number of characters in the text string must be less than or equal to 15. If you don’t want text on the button, you must specify two double quotes (“”) with no intervening characters.

Example

```
Local ( Result );

Result := Prompt ( “WARNING”, “Delete consolidation
results?”, “Yes”, “No” );

If ( Result = 1 )

Delete ( );
```

Creates a prompt panel that will display the specified text and two buttons and, depending on the user’s choice, causes the macro to take the appropriate action.

Argument	Description
Filename	Specifies the filename and path name (if necessary) of the spreadsheet to be printed. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename and path name must be surrounded by quotation marks. If you don't provide a filename, ParaSheet prints one copy of the spreadsheet that is the main window.

Examples

PrintNow ("Resources"); prints one copy of a spreadsheet named Resources (stored in the current folder) without showing the Print panel.

PrintNow ("/me/Data/Spreadsheets/ProjectFunds"); prints one copy of a spreadsheet called ProjectFunds, stored in another folder.

Prompt

Prompt (*Title, Message, Button1* [, *Button2* [, *Button3*]])

The *Prompt* statement causes a macro to stop running and displays a Prompt panel with a custom title and message. The Prompt panel can have up to three custom buttons. Prompt returns a value associated with the button used to dismiss the panel (see table below), and this information can then be used in calculations.

Returns	Description
1	If the user clicks Button 1 (the default)
0	If the user clicks Button 2
–1	If the user clicks Button 3

Argument	Description
Title	Specifies a title for the panel. The number of characters in the title must be less than or equal to 15, surrounded by quotation marks. If you don't want a title, you must

Examples

ChartSelect ("ChartText", 1, 1, "OilConsumption");

TextSelect (1, 100, 1, 100);

PointSize (16);

Sizes text in the specified chart text object at 16 points, assuming there are fewer than 100 words total and fewer than 100 characters in the last word.

Prefs

Prefs ( )

The *Prefs* command displays the ParaSheet Preferences panel, making its options available to the person running a macro. You must dismiss the panel for the macro to continue running. This command is equivalent to the Preferences command in the Info menu.

There are no arguments for *Prefs*, but parentheses must be included. *Prefs* will not execute in a user-defined function run from a cell.

Print

Print ( )

The *Print* command brings up the Print panel. You must dismiss the panel for the macro to continue running. This command is equivalent to the *Print* command in the main menu.

There are no arguments for Print, but parentheses must be included.

PrintNow

PrintNow ([*Filename*])

The *PrintNow* command activates the specified spreadsheet and causes one copy to print, bypassing the Print panel. This command is equivalent to printing with default settings in the Print panel (the *Print* command in the main menu).

Rate	A number specifying the periodic interest rate
Term	A number specifying the number of payment periods

Examples

Pmt (1000, 10% / 12, 12) gives $87.92 as the monthly payment needed to pay off a 12-month loan of $1000 at 10% interest.

Pmt (D2, C6 / 12, F4) gives $190.22 as the payment amount, when cell D2 contains 7500 (the principal), C6 contains .10 (the annual interest rate), and F4 contains the value 48 (the number of payments).

PointSize

PointSize (*Setting* [, *Selection*])

The *PointSize* command changes the font size of the contents of a selected cell or range of cells, chart text object, or chart label. This command is equivalent to settings in the Size text box in the Font panel.

Argument	Description
Setting	An integer specifying a new font size for the selected text. The minimum setting is 1 point, and the maximum setting is 255 points.
Selection	Specifies a cell, range of cells, or named range. If you don't provide a selection, ParaSheet changes the font size of the current selection.

To change the point size of the contents of a chart text object, you must first select the chart text object using the *ChartSelect* command. Then, use the *TextSelect* command to select its contents. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object or label are already selected.)

Plain

Plain ([*Selection* **])**

The *Plain* command makes the contents of selected cells, or chart text objects, plain, unstyled characters.

Argument	Description
Selection	Specifies a cell, range of cells, or named range. If you don't provide a selection, ParaSheet makes the current selection plain.

To make the contents of a chart text object plain, you must first select the chart text object using the *ChartSelect* command. Then, use the *TextSelect* command to select its contents. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object are already selected.)

Examples

```
ChartSelect ( "ChartText", 1, 1, "OilConsumption" );
TextSelect ( 1, 100, 1, 100 );
Plain ( );
```

Makes the contents of the chart text object in the chart named OilConsumption plain, assuming it has fewer than 100 words and fewer than 100 characters in the last word.

Pmt (Payment)

Pmt (***Principal, Rate, Term*** **)**

The *Pmt* function calculates the size of periodic payments necessary to pay off a loan at a periodic interest rate, over the number of payment periods in a specified term using the following formula:

$Principal * (Rate / 1 - (Rate + 1)^{-Term}$

Pmt assumes payments are made at the end of each period. If payments are made at the beginning of each period, divide the result by (1 + *Rate*).

Argument	Description
Principal	A number specifying the amount of the loan (the principal)

	"Yes" or 1
	"No" or 0
PieChartID	Specifies the name or number of the pie chart. If you don't provide a *PieChartID*, ParaSheet applies marking settings to the currently selected pie chart.

Examples

PieMarksSet ("Yes", "Yes", "No", "Yes", "OilConsumption"); displays the percentage values and legend categories for a pie chart named OilConsumption.

PieMarksSet (0, 0, 1, 1, 5); displays no marks for pie chart number 5, because the first argument (ShowLabels) specifies "No" and overrides the second, third, and fourth arguments.

PieStartAngleSet

PieStartAngleSet (*Angle* [, *PieChartID*])

The *PieStartAngleSet* command sets the starting angle for the first slice (the first plotted value) of a pie chart. *PieStartAngleSet* is only applicable to pie charts. This command is equivalent to the Starting Angle setting in the View Options subpanel in the Chart Inspector.

Argument	Description
Angle	A number between 0 and 360 (degrees) specifying the angle from which to start drawing the pie chart (0 is at the top of the circle).
PieChartID	Specifies the name or number of the pie chart. If you don't provide a *PieChartID*, ParaSheet resets the starting angle of the currently selected pie chart.

Example

PieStartAngleSet (270, "Wildlife"); resets the starting angle at 270 degrees (9 o'clock).

Examples

Pi ( ) gives 3.1416 in a cell formatted for four decimal places.

D2R (Sin (30)) gives 0.5, the sine of 30 degrees.

Sin (30 * Pi ( ) / 180) gives 0.5, the sine of 30 degrees.

Pi ( ) * 3 ^ 2 gives 28.274334, the area of a circle with a radius of 3.

PieMarksSet

PieMarksSet (*ShowLabels, Percent, Value, Category* [, *PieChartID*])

The *PieMarksSet* command determines what labels mark each slice of a pie chart. PieMarksSet is only applicable to pie charts. This command is equivalent to settings in the Pie Labels area of the Marking subpanel in the Chart Inspector.

Argument	Description
ShowLabels	Controls whether percentage, value, or category labels show. *ShowLabels* overrides entries for *Percent*, *Value*, and *Category*. Acceptable entries are: "Yes" or 1 "No" or 0
Percent	Specifies whether the percentage value for each pie slice shows. Acceptable entries are: "Yes" or 1 "No" or 0
Value	Specifies whether the numeric value for each pie slice shows. Acceptable entries are: "Yes" or 1 "No" or 0
Category	Specifies whether the legend categories for each pie slice show. Acceptable entries are:

PasteSpecial will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

PasteSpecial ("Yes", "Values", "Overwrite", "No", "No", A2:B6); Replaces the values in cells A2 through B6 with the contents of the pasteboard, including values and formatting but not formulas, and ignoring cell references.

PasteSpecial (0, "Both", 2, 0, 1); copies formulas and values but not formatting and adds them to the values in the current selection, preserving cell references.

PasteSpecial ( ); brings up the PasteSpecial panel.

Pause

Pause ( )

The *Pause* statement causes a macro to stop running and displays the Pause panel, with a standard message *Click Continue to continue running the macro* and a *Continue* button. Once the user responds to the Pause panel, the macro continues running.

There are no arguments for Pause, but parentheses must be included.

See Also

For more information on the Pause statement, see the "Making Macros Interactive" section on page 35 of this manual.

Pi

Pi ( )

The *Pi* function returns the value of pi (to a maximum of 14 digits), 3.1415926535897, which is the ratio of the circumference of a circle to its diameter. To convert degrees into radians for the trigonometric functions, you can use the D2R function or multiply the angle in degrees by pi/180, as shown in the example below.

There are no arguments for Pi, but parentheses must be included.

	pasted into the new location. Acceptable entries are: "Yes" or 1 to paste all formats "No" or 0 to ignore all formats
Contents	Specifies whether cell contents are pasted into the new location. Acceptable entries are: "None" or 0 to paste none of the contents "Formulas" or 1 "Values" or 2
Effect	Specifies how the pasted contents affect any existing values in the new location. Acceptable entries are: "Overwrite" or 1 "Add" or 2 "Subtract" or 3 "Multiply" or 4 "Divide" or 5
SelectedCells	Useful when the selection contains fewer cells than the number of cells on the pasteboard. If so, data is pasted into the selection only. Acceptable entries are: "Yes" or 1 "No" or 0
AbsoluteRef	Specifies whether original cell references remain the same in the new location or are updated. Acceptable entries are: "Yes" or 1 "No" or 0
Selection	Specifies a cell, range of cells or named range. If you don't provide a selection, ParaSheet pastes the current selection.

Examples

Paste (D5:E12); pastes the contents of the pasteboard into cells D5 through E12.

Cut (A1:C10);

Paste (EstuaryData);

Cuts data from cells A1 through C10 and pastes it into a named range called EstuaryData.

Local (ObjName);

ObjName := GetObjName (3);

Cut (3);

Paste (C1);

ObjInfo (ObjName);

Cut ("GrayOval");

Paste ("BlackOval");

Assume you have an oval named BlackOval and it is object number 3. This macro assigns the object name BlackOval to a variable, then cuts the oval and pastes it at a new location. ParaSheet assigns the oval a new object number. Then, this macro reassigns the name BlackOval to the oval, cuts another oval named GrayOval, and pastes it over BlackOval. GrayOval is placed 15 pixels below and to the right of BlackOval.

PasteSpecial

PasteSpecial ([*Formats, Contents, Effect, SelectedCells, AbsoluteRef* [, *Selection*]])

The *PasteSpecial* command inserts the specified contents of a range of cells (which can be the chosen combination of values, formats, and formulas) into a new location. Any absolute, relative, or mixed cell references are updated. This command is equivalent to choices in the Paste Special panel (the *Paste Special* command in the Edit menu).

Argument	Description
Formats	Specifies whether formatting attributes (like currency, decimal, bold, and so on) are

PageLayout

PageLayout ( )

The *PageLayout* command displays the ParaSheet Page Layout panel, so you can specify page layout options. This command is equivalent to the *Page Layout* command in the Format menu.

There are no arguments for PageLayout, but parentheses must be included.

Paste

Paste ([*Selection*])

The *Paste* command inserts the contents of the pasteboard into a cell, range of cells, or chart text object. Any absolute, relative, or mixed cell references are updated. If the pasteboard contents are a chart object, ParaSheet pastes the object at the selection you provide, aligning the top left corner of the object with the top left corner of the selection. The pasted object is given a new object number; if it was named you must rename it if you want to refer to it by name. This command is equivalent to the *Paste* command in the Edit menu.

Argument	Description
Selection	Specifies a cell, range of cells, or named range. If you are pasting an object, you can optionally specify an object name or number. The pasted object will be placed 15 pixels below and to the right of the top left corner of the specified object. If you don't provide a selection, ParaSheet inserts the contents of the pasteboard at the current selection.

To paste the contents of the pasteboard into a chart text object, you must first select the chart text object using the *ChartSelect* command. Then, use the *TextSelect* command to position the insertion point or select the contents you want to paste over. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object, or label are already selected.)

Paste will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

removes a page break at the current location.

Examples

PageBreakSet ("Yes", "Row"); sets a page break at the top row of the current selection.

PageBreakSet ("No", 2, D1); removes a page break at column D.

PageBreakSet (1, 1, AirQuality); sets a page break at the top of a named range called AirQuality.

PageBreakSet ("No", "All"); removes all manual page breaks from the current spreadsheet.

PageBreaksShow

PageBreaksShow (*Setting*)

The *PageBreaksShow* command specifies whether page breaks display. This command is equivalent to the *Show Page Breaks/Hide Page Breaks* commands in the Layout submenu.

Argument	Description
Setting	Specifies whether page breaks show in the spreadsheet window. Acceptable entries are: "Yes" or 1 "No" or 0

Examples

PageBreaksShow ("Yes"); causes page break indicators in the current spreadsheet to appear.

PageBreaksShow (0); hides the page break indicators in the current spreadsheet.

```
PageBreaksShow ( 1 );
Select ( D:D );
PageBreakSet ( "Yes", "Col");
```

Displays page breaks, selects column D, and sets a new page break.

OpenLinks will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

OpenLinks ("RegionalSales"); makes the spreadsheet named RegionalSales in the current folder the main window and brings up the OpenLinks panel.

OpenLinks ("/me/Data/Spreadsheets/NorthernOffice"); makes the spreadsheet named NorthernOffice in another folder the main window, changes the working folder, and brings up the OpenLinks panel.

PageBreakSet

PageBreakSet (*Setting, Type* [, *Selection*])

The *PageBreakSet* command sets a new page break below the selected row or to the right of the selected column, or removes an existing page break at a row or column. *PageBreakSet* can also remove all manual page breaks from a spreadsheet. This command is equivalent to the *Set Page Break, Remove Page Break,* or *Remove All Breaks* commands in the Sheet menu.

Argument	Description
Setting	Specifies whether a page break exists at the selection. Acceptable entries are: "Yes" or 1 to set a page break "No" or 0 to remove an existing page break (or all manual page breaks)
Type	Specifies whether the page break is added to or removed from a row or a column, or if all manual page breaks are removed. Acceptable entries are: "Row" or 1 "Col" or 2 "All" or 3 to remove all page breaks (if *Setting* is "No")
Selection	Specifies a column or row. If you don't provide a selection, ParaSheet adds or

must be surrounded by quotation marks. You can specify a unique path name for each spreadsheet you want to open.

Open will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

Open ( ); displays the Open panel.

Open ("Resources", "/me/Data/People"); opens two spreadsheets: one named Resources in the current folder, and another named People from the /me/Data folder.

Open ("/me/Data/Spreadsheets/DrinkingWater"); opens a spreadsheet named DrinkingWater in the /me/Data/Spreadsheets folder.

OpenLinks

OpenLinks ([*Filename*])

The *OpenLinks* command makes a specified spreadsheet the main window, then brings up the Open Links panel so you can choose a spreadsheet linked to the main spreadsheet window. If you don't specify a filename, the Open Links panel displays spreadsheets linked to the current spreadsheet. This command is equivalent to choosing the *Open Links* command from the Document menu and selecting a file from the Open Links panel.

Argument	Description
Filename	Specifies the filename and path name (if necessary) of the spreadsheet for which you want to open linked files. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename and path name must be surrounded by quotation marks.

VOffset	A positive or negative integer representing the number of cells to track vertically. A positive integer locates a cell below the current selection; a negative integer locates a cell above. If *VOffset* is 0, Offset returns the value in the top left cell of the specified range.

Examples

Offset (A3, 2, 5) displays the value in C8.

GetRowNum (Offset (G15, –3, –2)) gives 13, the index of the row of cell D13.

Sum (Offset (B5:C6, 1, 1)) gives the sum of the values in the range C6:D7. The Offset function here returns the range C6:D7, which is the range that's offset 1 cell horizontally and 1 cell vertically from B5:C6. This range is then used as an argument in the Sum function.

Open

Open ([*FirstFile* [, ... , *LastFile*]])

The *Open* command opens one or more existing spreadsheets. This command is equivalent to opening the Open panel (using the *Open* command in the Document menu) and selecting one or more files.

Argument	Description
FirstFile	Specifies the filename and path name (if necessary) of the first spreadsheet to be opened. If you don't provide a path name, ParaSheet assumes the file is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename and path name must be surrounded by quotation marks.
LastFile	Specifies the filenames and path names (if necessary) of additional spreadsheets to be opened. If you don't provide a path name, ParaSheet assumes the file is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename and path name

ObjSize

ObjSize (*NewWidth, NewHeight*)

The *ObjSize* command establishes a new size for the selected object in the spreadsheet.

Argument	Description
NewWidth	A positive integer specifying the amount (in pixels) by which to resize the object horizontally
NewHeight	A positive integer specifying the amount (in pixels) by which to resize the object vertically

Example

ObjSize (120, 140); resizes the currently selected object to 120 pixels wide by 140 pixels tall.

Offset

Offset (*Range, HOffset, VOffset*)

The *Offset* command returns a range reference a specified distance from the top left cell in the provided range. If you use *Offset* alone, the value is displayed in the cell. You can also use *Offset* to provide an argument to a function expecting a range reference — for example, as an argument for the *GetRowNum* command and the *Sum* function (see examples, below).

Argument	Description
Range	Specifies a cell, range of cells, or named range. If you specify a range of cells or named range, Offset moves horizontally and vertically from the top left cell in that range.
HOffset	A positive or negative integer representing the number of cells to track horizontally. A positive integer locates a cell to the right of the current selection; a negative integer locates a cell to the left. If *HOffset* is 0, Offset returns the value in the top left cell of the specified range.

	in the spreadsheet or chart. Negative integers move left and positive integers move right.
YDirection	An integer specifying the amount (in pixels) to move an object or chart part vertically in the spreadsheet or chart. Negative integers move up and positive integers move down.
Reference	Specifies a location from which the object or chart part is to be moved. Acceptable entries are: A cell A range of cells A named range An object name or number If you specify a range or an object name or number, ParaSheet uses the top left cell under the range or object as a reference point. If the object being moved is within a chart, ParaSheet ignores the *Reference* argument. If you don't provide an argument for Reference, the object is moved horizontally and vertically from its current location.

Examples

Select ("TitlePara");

ObjMove (39, 16, C1);

Selects the object named TitlePara and moves it 39 pixels to the right and 16 pixels below the top left corner of cell C1.

SelectAdd ("Pointer", "TitlePara", "Disclaimer");

ObjMove (184, –92);

Selects three objects named Pointer, TitlePara, and Disclaimer and moves them 184 pixels (2 inches) to the right and 92 pixels (1 inch) up.

ChartSelect ("DataSeries", 2, 1, 1);

ObjMove (23, –23);

Selects the second pie wedge in chart number 1 and moves it 23 pixels right and 23 pixels up. This is an example of exploding a pie chart.

The symbol **&** *is the operator for concatenation.*

ObjInfo

ObjInfo (*ObjName*)

The *ObjInfo* command gives an object a descriptive name that you specify, and optionally protects the object from being moved or changed. This command is equivalent to options in the Common subpanel in the Chart Inspector).

Argument	Description
ObjName	Specifies a descriptive name for the object. The object name must be less than 256 characters long, and surrounded by quotation marks.
Locked	Specifies whether the object is protected from change. Acceptable entries are: "Yes" or 1 (to lock the object) "No" or 0 (to unlock the object) If you don't provide a lock setting, the object remains at its current setting.

Example

```
Select ( 5 );
ObjInfo ( "Pointer", "Yes" );
```

Selects object number 5 (a chart), gives it the name Pointer, and locks it.

ObjMove

ObjMove (*XDirection, YDirection* [, *Reference*])

The *ObjMove* command moves the selected object to a new location in the spreadsheet. It also moves the legend, graph, or a chart text object to a new location in a chart.

Argument	Description
XDirection	An integer specifying the amount (in pixels) to move the object or chart part horizontally

Acceptable entries are:

1 for General

2 for Currency

3 for Percent

4 for Fixed

5 for Scientific

6 for Logical

If you don't specify a format, ParaSheet uses the General format. The formats listed above are for the purpose of formatting text strings only, and have no correspondence with formats in the ParaSheet Format Inspector.

DecPlaces — A number specifying the number of decimal places to display in the returned string. You do not have to include the number of decimal places when Format is 6 (Logical) — if *DecPlaces* is included in this case, it will be ignored. If *DecPlaces* is not specified, no decimal places are displayed — except when the string is displayed in General format. In this case, ParaSheet displays up to six places to the right of the decimal point.

Commas — Specifies whether commas are inserted in the text string to the left of every set of three digits to the left of the decimal place.

Acceptable entries are:

"True" or 1 to insert commas

"False" or 0 for no commas

If *Commas* is not specified, ParaSheet doesn't insert any commas.

Examples

Num2C (1234, 2, 2, "True") returns the string "$1,234.00".

Num2C (24.4, 3, 0) returns the string "2440%".

Num2C (5.2521, 4, 1) & " million" returns the string "5.3 million".

"The population is " & Num2C (Sum (K1:K8)) & " thousand" returns "The population is 623 thousand" when the sum of cells K1 through K8 is 623.

NPV (10%, A1:A5) –80000 gives $17,848.20 as the net present value of the flows in cells A1 through A5, with an initial investment of $80,000 at an interest rate of 10%.

NRows

NRows ([*Range*])

The *NRows* command returns a positive integer representing the total number of spreadsheet rows in a range. If you don't provide an argument, *NRows* returns the number of rows in the current selection.

Argument	Description
Range	Specifies any range of cells or named range

Examples

NRows (A2:G38); returns 37.

NRows (January); returns 12 if January is a named range reference for B1:P12.

Num2C (Number to Text String)

Num2C (*Number* [, *Format* [, *DecPlaces* [, *Commas*]]])

The *Num2C* function converts the numeric value of a specified number into a text string with the specified format and number, decimal places, and commas. *Num2C* is useful for entering a computed value as part of a text string in a cell.

Argument	Description
Number	Specifies the numeric value to be converted to a text string. Acceptable entries are: A numeric value (either a constant or a formula that evaluates to a numeric value) A cell or range that contains a numeric value
Format	Specifies a number representing the type of format to be assigned to the string.

NPV (Net Present Value)

NPV (*Rate, Flows*)

The *NPV* function returns the net present value of future cash flows, discounted at a periodic interest rate, using the formula:

$\Sigma[f_i / (1 + Rate)^i]$

where the f_i are the cash flows. ParaSheet assumes that the cash flows occur at equal time intervals, with the first flow occurring at the end of the first period. If there is an initial investment at the beginning of the first period, you must add it separately, as shown in the example below.

Argument	Description
Rate	Specifies the periodic interest rate at which the flows are discounted
Flows	Specifies the list of the cash flow amounts. Negative values represent cash outflows, and positive values represent cash receipts. Acceptable entries are: Numeric values Cells or ranges containing numeric values A combination of the above

Examples

	A	B	C
1	$5,000.00		$50.00
2	$ 15,000.00		$50.00
3	$30,000.00		$50.00
4	$40,000.00		$50.00
5	$50,000.00		$50.00
6			$50.00
7			$50.00
8			$50.00
9			$50.00
10			$50.00
11			$50.00
12			$50.00

NPV (10%, C1:C12) gives $340.68 as the net present value of the flows in cells C1 through C12 at an interest rate of 10%.

Argument	Description
XDirection	A positive or negative integer specifying the number of cells to move horizontally. A positive integer selects a cell to the right of the current selection; a negative integer selects a cell to the left.
YDirection	A positive or negative integer specifying the number of cells to move vertically. A positive integer selects a cell below the current selection; a negative integer selects a cell above.

Examples

NextCell (3, 5); selects the cell three cells to the right and five cells below the current selection.

NextCell (–3, –7); selects the cell three cells to the left and seven cells above the current selection.

NextCell (Horiz, Vert); selects the cell directly below the current cell, when the variable Horiz equals 0 and the variable Vert equals 1.

Now (Current Date and Time)

Now ( )

The *Now* function returns the date number of the current date and time. The value of *Now* is updated every time the spreadsheet is recalculated. (See the "Time Functions" section on page 54 of this manual.)

There are no arguments for *Now,* but parentheses must be included.

Examples

If the present date were March 2, 1992, and the time were 12:24:18 PM:

Now ( ) would give 32203.516875.

Month (Now ( )) would give 3.

Minute (Now ( )) would give 24.

NCols

NCols ([*Range*])

The *NCols* command returns a positive integer representing the total number of spreadsheet columns in a range. If you don't provide an argument, *NCols* returns the number of columns in the current selection.

Argument	Description
Range	Specifies any range of cells or named range

Examples

NCols (B5:F10); gives 5.

NCols (Sales); gives 15 if Sales is a named range reference for B1:P12.

New

New ( )

The *New* command opens a new spreadsheet window and makes it the main window. This command is equivalent to the *New* command in the Document menu.

There are no arguments for New, but parentheses must be included.

New will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Example

New ( ); creates a new, untitled spreadsheet.

NextCell

NextCell (*XDirection, YDirection*)

The *NextCell* command selects a cell that is the specified distance from the currently selected cell. If the current selection is an object, ParaSheet selects the cell that is the specified distance from the cell at the top left corner of the object.

NamedRngEdit

NamedRngEdit (*OldName, NewName* [, *NewRange*])

The *NamedRngEdit* command changes the specified range name from its current name to a new name. This command is equivalent to using the *Modify* button in the Named Ranges panel (the *Named Ranges* command in the Sheet menu).

Argument	Description
OldName	Specifies the name of an existing named range. The range name must be surrounded by quotation marks.
NewName	Specifies the new name to be assigned to the named range. The range name must consist of fewer than 246 alphanumeric characters and underscores, and be surrounded by quotation marks.
NewRange	A range reference specifying the new boundaries of the range. If you don't specify a new range, ParaSheet applies the new name to the existing range.

NamedRngEdit will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

NamedRngEdit ("WaterLevels", "ReservoirLevels"); changes the range name WaterLevels to ReservoirLevels for the selected range.

Select (WaterLevels);

NamedRngEdit ("WaterLevels", "ReservoirLevels", D2:D24);

Selects the named range called WaterLevels (D2 through D12), changes the name of the range to ReservoirLevels, and increases the range by 12 cells.

	and underscores, and be surrounded by quotation marks.
Selection	Specifies the cell or range of cells to be named. If you don't provide a selection, ParaSheet applies the name to the currently selected range.

NamedRngAdd will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

NamedRngAdd ("WaterLevels"); assigns the currently selected cells to a named range called WaterLevels.

NamedRngAdd ("Results1", A2:A12); assigns cells A2 through A12 to a named range called Results1.

NamedRngDel

NamedRngDel (*RangeName*)

The *NamedRngDel* command removes the specified range name from the data with which it is associated. This command is equivalent to using the *Delete* button in the Named Ranges panel (the *Named Ranges* command in the Sheet menu).

Argument	Description
RangeName	Specifies the name to be removed from a range of cells. The range name must be surrounded by quotation marks.

NamedRngDel will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Example

NamedRngDel ("WaterLevels"); deletes the range name WaterLevels from the selected range.

Example

NamedRngAdd ("Results1", A2:A12);

NamedRngAdd ("Results2", C2:C12);

SelectAdd (Results1, Results2);

Move (Results1, Results2);

Creates two named ranges, makes a multiple selection, and copies data from the first range to the second.

NA (Not Available Value)

NA ( )

The *NA* function assigns the value !N/A (value-not-available error) to any cell in which it is entered. This function can act as a placeholder in the spreadsheet when you are unable to enter a value needed to complete a formula. Any cell with a formula dependent upon a cell that contains !N/A will display !N/A. This prevents the occurrence of a chain of inaccurate calculations throughout a spreadsheet.

There are no arguments for *NA*, but parentheses must be included.

Examples

NA ( ) gives !N/A in cell E5 when it's entered in cell E5.

Sum (E1:E6) gives !N/A when cell E5 contains !N/A, because this formula is dependent on cell E5.

NamedRngAdd

NamedRngAdd (*RangeName* [, *Selection*])

The *NamedRngAdd* command gives a name to the specified range of cells. This command is equivalent to using the *Add* button in the Named Ranges panel (the *Named Ranges* command in the Sheet menu).

Argument	Description
RangeName	Specifies the name to be assigned to a range of cells. The range name must consist of less than 246 alphanumeric characters

Month (Month of the Year)

Month (*DNumber*)

The *Month* function returns an integer from 1 to 12 (inclusive), corresponding to the month of the year for the date number you specify. 1 represents January, 12 represents December.

Argument	Description
DNumber	Specifies a valid date number (see the "Time Functions" section page 54 of this manual)

Examples

Month (37903) gives 10 (October).

Month (Date (93, 1, 5)) gives 1 (January). The Date function here gives the date number corresponding to 1/5/93, which is then used as the specified date number for the Month function.

Move

Move ([*Source, Destination*])

The *Move* command takes the selected data from its current location and puts it in a new location. If you don't specify any arguments but have two separate cells or ranges selected, ParaSheet moves data from the first to the second selection.

(Move does not work with multiple selections of more than two ranges.) Any absolute, relative, or mixed cell references are updated. This command is equivalent to the *Move Cells* command in the Cells submenu.

Argument	Description
Source	Specifies the cell, range of cells, or named range containing data to be moved
Destination	Specifies the cell, range of cells, or named range to which the data will be moved

Move will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

The returned integer will be between 0 and the base number, excluding the base number.

Argument	Description
Number	Specifies a numeric value. Acceptable entries are: A number (constant or formula) A cell that contains a number A range, the first cell of which contains a numeric value
Base	Specifies the number to be divided into the specified *Number*. Acceptable entries are: A nonzero number A formula that evaluates to a nonzero number A cell that contains a nonzero number A range, the first cell of which contains a nonzero number The following limitations apply: If *Base* = 0, then Mod gives a !NUM error If *Base* < 0, then Mod gives a value <= 0, but > – *Base* If *Base* > 0, then Mod gives a value >= 0, but < *Base*

Examples

Mod (15, 6) gives 3, the remainder of 15 divided by 6.

Mod (15, –6) gives –3.

Mod (–15, 4) gives 1.

Mod (–15, –4) gives –3.

Mod (5, 15) gives 5.

than the current working folder, ParaSheet changes the working folder to the new path. The filename (including path name) must be surrounded by quotation marks. If you don't provide a Filename, ParaSheet maximizes the currently selected spreadsheet.

Examples

Miniaturize ("Resources"); causes the open spreadsheet named Resources (stored in the current folder) to appear as an icon in the workspace.

Miniaturize ("/me/Data/Spreadsheets/Resources"); causes the open spreadsheet named Resources (stored in another folder) to appear as an icon in the workspace.

Minute (Minute of the Hour)

Minute (*DNumber*)

The *Minute* function returns an integer from 0 to 59 (inclusive), corresponding to the minute of the hour for the date number you specify. 0 is on the hour, 1 is one minute after the hour, and 59 is one minute before the next hour.

Argument	Description
DNumber	Specifies a valid date number (see page 54 of this manual for more on Time functions).

Example

Minute (0.7) gives 48.

Mod (Numeric Modulus)

Mod (*Number, Base*)

The *Mod* function returns the remainder of a specified number after it has been divided by a base number, according to the following formula:

Number – Floor (*Number* / *Base*) * *Base*

where Floor (*Number* / *Base*) is the largest integer <= the value of (*Number* / *Base*).

Min (Minimum Value)

Min (*List*)

The *Min* function returns the minimum of the numeric values in a list.

Argument	Description
List	Specifies a list of one or more values. The items in the list must be separated by commas. Acceptable entries are: A numeric value or values (formulas or constants) A text value or values Null values Cells or ranges containing numeric, text, or null values Any combination of the above Min ignores text and null values.

Examples

Min (500, 250, 57, 320) gives 57.

Min (C1, 45, 32, 2 + 13, "stop") gives 15, when cell C1 contains 50.

Miniaturize

Miniaturize ([*Filename*])

The *Miniaturize* command causes a spreadsheet window to display as an icon in the workspace. This command is equivalent to the *Miniaturize Window* command in the Windows menu.

Argument	Description
Filename	Specifies the name of the spreadsheet to be miniaturized. If the spreadsheet is in another folder, you must provide a path name. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other

	A cell that contains a text string
	A range, the first cell of which contains a text string
	A formula that evaluates to a text string
StartChar	Specifies the position at which to start extracting characters. 1 indicates the first character in the string, 2 indicates the second, and so on.
	Acceptable entries are:
	A positive, nonzero integer between 1 and the total number of characters in the string, inclusive
	A formula, cell, or range that evaluates to a positive nonzero integer between 1 and the total number of characters in the string, inclusive
NumChars	Specifies the number of characters to be returned. Acceptable entries are:
	A non-negative integer
	A formula that evaluates to a non-negative integer
	A cell or range that contains a non-negative integer value
	When the value of *NumChars* exceeds the total number of characters from the starting character to the end of the string, Mid returns all the characters from the starting character to the end of the string.

Examples

Mid (H16, 9, 7) gives "Affairs" when cell H16 contains "Foreign Affairs".

Mid ("National Headquarters", 10, 15) gives "Headquarters".

Maximize

Maximize ([*Filename*])

The *Maximize* command redisplays a miniaturized spreadsheet as a spreadsheet window.

Argument	Description
Filename	Specifies the name of the spreadsheet to be maximized.If the spreadsheet is in another folder, you must provide a path name. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename (including path name) must be surrounded by quotation marks. If you don't provide a Filename, ParaSheet maximizes the currently selected spreadsheet.

Examples

Maximize ("RecycleFactors"); causes the miniaturized spreadsheet named RecycleFactors to appear as a spreadsheet window.

Maximize ("/me/Data/Spreadsheets/QuarterlySales"); causes the specified spreadsheet (which is miniaturized) to appear as a spreadsheet window.

Mid (Extract a Substring)

Mid (*String, StartChar, NumChars*)

The *Mid* function returns a specified number of characters (including spaces) from a text string, beginning with the character in a specified position.

Argument	Description
String	Specifies the text string. Acceptable entries are: A sequence of ASCII characters surrounded by quotation marks

Max (Maximum Value)

Max (*List*)

The *Max* function returns the maximum of the numeric values in a list.

Argument	Description
List	Specifies a list of one or more values. The items in the list must be separated by commas. Acceptable entries are: A numeric value or values (formulas or constants) A text value or values Null values Cells or ranges containing numeric, text, or null values Any combination of the above Max ignores text and null values.

Examples

	A	B
1	251	30
2	102	68
3	220	440
4	38	234
5	117	267
6	114	480
7	83	133
8	191	112

Max (A1:A8) gives 251.

Max (B4:B8, 250, A8, –50, 2003) gives 2003.

Examples

Local (GlassWaste, PaperWaste := 25, MetalWaste); declares three variables using the specified names and assigns the value 25 to PaperWaste. Values can be assigned to GlassWaste and MetalWaste later in the macro.

Local (Disclaimer := "This total may be slightly inaccurate because of missing data."); assigns a text string to the variable named Disclaimer.

Local (Results := Avg (UnitsShipped)); assigns the average of a named range called UnitsShipped to the local variable Results.

See Also

For more information on variables, see the "Using Variables" section on page 20 of this manual.

Log (Log Base 10)

Log (*Number*)

The *Log* function returns the common logarithm, or log base 10, of a specified number.

Argument	Description
Number	Specifies the number for which ParaSheet returns the common logarithm. Acceptable entries are: A numeric value (formula or constant) > 0 A cell that contains a numeric value > 0 A range, the first cell of which contains a numeric value > 0

Examples

Log (2) gives 0.30103.

Log (4) gives 0.60206.

Log (100) gives 2.

Ln (Natural Log)

Ln (*Number*)

The *Ln* function returns the natural logarithm, or log base *e*, of a specified number. *Ln* is the inverse of the Exp function.

Argument	Description
Number	Specifies the number for which ParaSheet returns the natural logarithm. Acceptable entries are: A numeric value (formula or constant) > 0 A cell containing a numeric value > 0 A range, the first cell of which contains a numeric value > 0

Examples

Ln (2) gives 0.693147.

Ln (4) gives 1.386294.

Ln (Exp (12)) gives 12.

Local

Local (*VariableList*)

The *Local* statement declares and optionally assigns values to one or more local variables; that is, variables that can only be accessed from the macro that contains them. If you don't assign a value to a local variable, it is initialized to the integer value 0. A local variable exists only while the macro in which it is declared is running.

Argument	Description
VariableList	Specifies one or more variables, each with an assigned value. Multiple variables must be separated by commas.

When the value of *NumChars* exceeds the total number of characters in the string, Left returns all the characters in the string.

Examples

Left ("Smith John", 8) gives "Smith Jo".

Left ("Transitional", D4) gives "Trans" when cell D4 contains 5.

Left ("Transitional", 45) gives "Transitional".

Left (Choose (1, "beginning", "middle", "end"), 3) gives "mid". The Choose function here returns the text string "middle", the second argument in its *List*. This string is then used as an argument in the Left function.

Length (Length of Text)

Length (*String*)

The *Length* function returns the number of characters (including spaces) in a text string.

Argument	Description
String	Specifies the text string. Acceptable entries are: A sequence of ASCII characters surrounded by quotation marks A cell that contains a text string A range, the first cell of which contains a text string A formula that evaluates to a text string

To italicize the contents of a chart text object or chart label, you must first select the chart text object. Then, use the *TextSelect* command to select its contents. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object or label are already selected.)

Examples

Select ("OilConsumption");

ChartSelect ("ChartText", 1, 1);

TextSelect (1, 1000)

Italic ("Yes");

Makes the contents of the chart text object in the chart named OilConsumption italic.

Left (Left Substring)

Left (*String, NumChars*)

The *Left* function returns a specified number of characters (including spaces) furthest left in a text string.

Argument	Description
String	Specifies the text string.
	Acceptable entries are:
	A sequence of ASCII characters surrounded by quotation marks
	A cell that contains a text string
	A range, the first cell of which contains a text string
	A formula that evaluates to a text string
NumChars	Specifies the number of characters to be returned. Acceptable entries are:
	A non-negative integer
	A formula that evaluates to a non-negative integer
	A cell or range that contains a non-negative integer value

Argument	Description
Expression	Specifies the expression to be tested. Acceptable entries are: A text value A numeric value (formula or constant) A cell or range reference

Examples

IsText (B1:D10) gives True (1) when cell B1 contains "Summary".

IsText (D1:D10) gives False (0) when cell D1 contains the numeric value 2014.

IsText (A39) gives False (0) when cell A39 is blank.

IsText (D29) gives False (0) when the formula in cell D29 evaluates to an error.

IfX (IsText (B9), B9, "No") gives the text in B9 when B9 contains text; otherwise, it gives "No". The IsText function evaluates whether the value in cell B9 is text. The result of this evaluation (either True or False) is then used as an argument in the IfX function.

Italic

Italic (*Setting* [, *Selection*])

The *Italic* command determines whether the contents of selected cells, or chart text objects are italic. This command is equivalent to the *Italic/Unitalic* commands in the Font submenu.

Argument	Description
Setting	Specifies whether characters are italic. Acceptable entries are: "Yes" or 1 to make characters italic "No" or 0
Selection	Specifies a cell, range of cells, or named range. If you don't provide a selection, ParaSheet italicizes the current selection.

Examples

IsNull (G3) gives False (0) if G3 contains a text or numeric value, an error value, or a reference to a non-empty cell or range.

IsNull (B13) gives True (1) if cell B13 is empty (not zero).

IsNum (Numeric Value Test)

IsNum (*Expression* **)**

The *IsNum* function tests whether a specified expression is a numeric value. If it is, IsNum returns 1 (or logical True). If it isn't, *IsNum* returns 0 (or logical False).

Argument	Description
Expression	Specifies the expression to be tested. Acceptable entries are: A text value A numeric value (formula or constant) A cell or range reference

Examples

IsNum (A3) gives False (0) if cell A3 contains the string "Western".

IsNum (C5) gives True (1) if cell C5 contains the value 112.88

IsNum (A6) gives False (0) if cell A6 contains a formula that evaluates to an error.

IsNum (D1:D5) gives False (0) if cell D1 is blank.

IsNum ("3.52") gives False (0), because "3.52" is a string, not a numeric value.

IsText (Text Value Test)

IsText (***Expression*** **)**

The *IsText* function tests whether a specified expression is a text value. If it is, IsText returns 1 (or logical True). If it isn't, *IsText* returns 0 (or logical False). Error codes are not considered text values.

Examples

	A	B	C	D
1		($ 1000.00)	($ 1000.00)	
2		$ 120.00	$ 120.00	
3		$ 120.00	$ 124.00	
4		$ 120.00	$ 128.00	
5		$ 120.00	$ 132.00	
6		$ 120.00	$ 136.00	
7		$ 120.00	$ 130.00	
8		$ 120.00	$ 120.00	
9		$ 120.00	$ 124.00	
10		$ 120.00	$ 128.00	
11		$ 120.00	$ 132.00	
12		$ 120.00	$ 120.00	
13		$ 120.00	$ 124.00	
14				

IRR (5%, B1:B13) calculates a 6.11% rate of return over a one-year period, beginning with 5% as the estimate for the rate. The initial investment is $1,000.00.

IRR (5%, C1:C13) gives a 7.09% rate of return over a one-year period, beginning with an estimate of 5%. The initial investment is $1,000.00.

IsNull (Null Value Test)

IsNull (*Expression*)

The *IsNull* function tests whether a specified expression is a null value. If it is, *IsNull* returns 1 (or logical True). If it isn't, IsNull returns 0 (or logical False).

Argument	Description
Expression	Specifies the expression to be tested. Acceptable entries are: A text value A numeric value (formula or constant) A cell or range reference

IRR (Internal Rate of Return)

IRR (*Estimate, Flows*)

The *IRR* function returns an approximate internal rate of return (the amount yielded) for a series of cash flows, such that the net present value of the cash flows is 0. The internal rate of return is calculated iteratively to 8-digit precision to achieve the rate for which:

$\Sigma[f_i / (1 + Rate)^{i}] = 0$

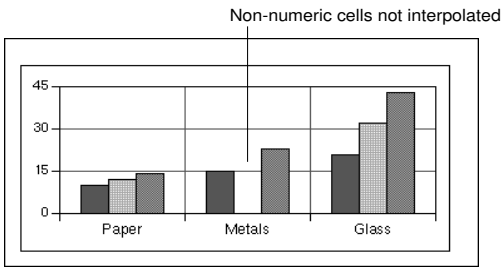
where the f_i represents the specified cash flows. ParaSheet begins computing the internal rate of return as the value of a specified estimate. It stops after achieving 8-digit precision, or after 20 iterations.

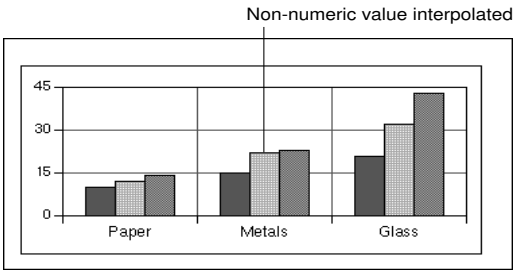
Argument	Description
A number	Represents the profitability of an investment over
(percentage)	a certain time period

Argument	Description
Estimate	Specifies an initial guess at the rate of return
Flows	Specifies a list of the cash flow amounts. The first of these should be the initial investment, if there is one. Negative values in represent cash outflows, and positive values represent cash receipts. Acceptable entries are a list of: Numeric values Cells or ranges containing numeric values A combination of the above

Example

	A	B	C
1	Paper	Metals	Glass
2	10	15	21
3	12	Not Available	32
4	14	23	43





ChartNew ("Bar", A2:C4);

InterpSet ("Yes");

Creates a bar chart using values in cells A2 through C4, then interpolates non-numeric cells (B3, in this example). Initially, the bar chart displays only two values for the metals category because interpolation is off. When non-numeric cells are interpolated, ParaSheet calculates the missing value for metals based on the existing values, then displays the third bar.

	A range, the first cell of which contains a numeric value

Examples

Int (42.765) gives 42.

Int (C6) gives –636 when cell C6 contains –636.214.

InterpSet

InterpSet (*Interpolate* [, *ChartID*])

The *InterpSet* command specifies whether to assign interpolated values to non-numeric cells with the data set being charted. If interpolation is off, non-numeric cells are charted as if their values were zero. This command is equivalent to the *Interpolate Non-numeric Cells* check box in the Chart Inspector. It is not applicable to high-low or pie charts.

Argument	Description
Interpolate	Specifies whether non-numeric cells are interpolated. Acceptable entries are: "Yes" or 1 to interpolate "No" or 0 to exclude non-numeric cells The ParaSheet default is not to interpolate.
ChartID	Specifies the name or number of a chart. If you don't provide a chart name or number, ParaSheet interpolates non-numeric cells into the currently selected chart.

	insertion. If you don't provide a replace direction, the Insert panel appears. Acceptable entries are: "Right" or 1 "Down" or 2
Selection	Specifies the cells, columns, or rows to be inserted. If you specify multiple columns or rows, ParaSheet inserts the same amount as you specify. Acceptable entries are: cell references, range names, column letters, row numbers. If you don't provide a selection, ParaSheet inserts one row or column at the current selection.

Insert will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

Insert ( ); brings up the Insert panel.

Insert (2, 5:5); inserts a new row at row 5, and moves the contents of row 5 down.

Insert ("Right", Sediment); inserts a range of cells the same size as the named range called Sediment, and moves the contents of Sediment to the right. Contents of cells below the named range Sediment remain in the same location.

Int (Integer Value)

Int (*Number*)

The *Int* function returns the integer part of a specified number. *Int* rounds up for negative numbers and down for positive numbers.

Argument	Description
Number	Specifies a numeric value. Acceptable entries are: A number (constant or formula) A cell that contains a numeric value

Examples

IncFill ("Columns", "Numeric", "Linear", 35, 5, Projection); fills cells in the named range called Projection as shown:

35	85	135
40	90	140
45	95	145
50	100	150
55	105	155
60	110	160
65	115	165
70	120	170
75	125	175
80	130	180

IncFill (1, 1, 1, 1, 10, A3:J3); fills cells A3 through J3 with the values 1, 11, 21, and so on.

IncFill (2, "Date", "Day", "1/2/1991", 1); fills the current selection with dates starting at 1/2/1991 and incrementing by one.

Info

Info ( )

The *Info* command causes the ParaSheet Info panel to appear. This command is equivalent to the *Info Panel* command in the Info menu.

There are no arguments for Info, but parentheses must be included.

Insert

Insert ([*ReplaceDirection* [, *Selection*]])

The *Insert* command adds cells, columns, or rows to the spreadsheet. Remaining columns move to the right and rows move down in the spreadsheet, depending on what was selected. Any absolute, relative, or mixed cell references are updated. This command is equivalent to choices in the Insert panel (the *Insert Cells* command in the Cells submenu).

Argument	Description
ReplaceDirection	Specifies the direction in which remaining cells, rows, or columns will move after the

	"Date" or 2
Value	If *Type* is Numeric, specifies whether the stepping value is added to or multiplied by the value in each preceding cell. If you don't provide a value, ParaSheet calculates in a linear fashion. If *Type* is Date, specifies the calendar time by which to calculate when the starting value is a date. If you don't provide a date value, ParaSheet calculates in days. Acceptable entries are: "Linear" or 1 (Step value is added) "Growth" or 2 (Step value is multiplied) "Day" or 3 "Weekday" or 4 "Month" or 5 "Year" or 6
Start	A number (integer or floating point) specifying the value in the top left cell of the selection. If you don't provide a starting value, ParaSheet starts with 0.
Step	A number (integer or floating point) specifying the value by which the starting value increments or decrements. If you don't provide a step value, ParaSheet doesn't increment or decrement, and the starting value fills each cell in the selection.
Selection	Specifies a range of cells or named range. If you don't provide a selection, ParaSheet fills the current selection.

IncFill will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

	marks. ParaSheet supports only Lotus 1-2-3 2.01 and earlier versions. Acceptable entries are: "Lotus" or 1 "Text" or 2 If you provide a *Filename* but not a *FileType*, ParaSheet imports the file as text.

Import will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

Import ("/me/Data/Spreadsheets/Energy.wk1", "Lotus"); opens a Lotus 1-2-3 file named Energy.

Import ("Resources", 2); opens a text file named Resources in the current folder.

IncFill

IncFill ([*Direction, Type, Value, Start, Step* [, *Selection*]])

The *IncFill* command inserts values into the selected cells by rows or by columns, starting with a specified value and increasing or decreasing by another specified value. Values can increment or decrement in either a linear or growth fashion. If you don't provide any arguments, the Incremental Fill panel appears. This command is equivalent to choices in the Incremental Fill panel (the *Incremental Fill* command in the Fill submenu).

Argument	Description
Direction	Specifies whether values are entered horizontally in rows or vertically in columns. Acceptable entries are: "Rows" or 1 "Columns" or 2
Type	Specifies whether the fill type is numeric or date. Acceptable entries are: "Numeric" or 1

	An error value
	A cell or range reference

Examples

IfX (5 < 6, B12, D12) displays the value in B12, because the expression 5 < 6 is True.

IfX (ABC < 25, "Sell", ABC) gives the string "Sell" when the value in the first cell of the named range ABC is < 25, and displays the value in ABC when it is >= 25.

IfX (Sales > 4000, Sales * Commission1, Sales * Commission2) checks to see if the value in the named range Sales is greater than 4000, multiplies it by the value in the named range Commission1 if it is greater, or multiplies it by the value in the range Commission2 if it isn't.

Sum (IfX (A7 = C7, A7:A10, E7:E10)) gives the sum of the values in the range A7:A10 when A7 = C7, and gives the sum of the values in the range E7:E10 when A7 <> C7.

Import

Import ([*Filename* [, *FileType*]])

The *Import* command opens an ASCII file or spreadsheet created in an application other than ParaSheet or on a computer running something other than NEXTSTEP. If you don't provide any arguments, the Import panel appears. This command is equivalent to choices in the Import panel (the *Import* command in the Document menu).

Argument	Description
Filename	Specifies the name of the file, including the extension, to be imported. If the file is in another folder, you must provide a path name. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename (including path name) must be surrounded by quotation marks.
FileType	Specifies the type of file to be imported. The file type must be surrounded by quotation

Examples

IfErr (E12, "WRONG", "OK") gives "WRONG" when cell E12 contains an error code; otherwise it gives "OK".

Sum (IfErr (B1, C1:C10, B1:B10)) gives the sum of the values in the range C1:C10 when cell B1 contains an error, and gives the sum of the values in the range B1:B10 when B1 doesn't contain an error.

IfX (Condition Evaluation)

IfX (*Condition, TrueVal, FalseVal*)

The *IfX* function tests whether a specified condition is true or false and returns TrueVal if it's true and FalseVal if it's false. This function should not be confused with the macro statement If.

When the result of the *IfX* function is a cell or range, *IfX* returns the range reference. Typically, this reference is used to display the value in the referenced cell or range. However, you can also use IfX wherever a range reference is appropriate (if ranges are used in the *TrueVal* and *FalseVal* arguments)—for example, as an argument for the *Sum* function, as shown below.

Argument	Description
Condition	Specifies an expression or logical condition that evaluates to True or False
TrueVal	Specifies the expression returned if the given condition evaluates to True. Acceptable entries are: A text value A numeric value (formula or constant) An error value A cell or range reference
FalseVal	Specifies the expression returned if the given condition evaluates to False. Acceptable entries are: A text value A numeric value (formula or constant)

Argument	Description
Expression	Specifies any valid expression to be tested. Acceptable entries are: Text A numeric value (formula or constant) An error value A cell or range reference
TrueVal	Specifies any valid expression to be returned if the given condition evaluates to True. Acceptable entries are: Text A numeric value (formula or constant) An error value A cell or range reference
FalseVal	Specifies any valid expression to be returned if the given condition evaluates to False. Acceptable entries are: Text A numeric value (formula or constant) An error value A cell or range reference When the result of the IfErr function is a cell or range, IfErr returns the range reference. ParaSheet can use this reference to display the value in the referenced cell or range, which is the usual result of IfErr. However, you can also use IfErr (with ranges for *TrueVal* and *FalseVal*) wherever a range reference is appropriate — for example, as an argument for the Sum function, as shown below.

CIRC errors cannot be compared using the IfErr function. !CIRC errors always propagate !CIRC when used in a formula.

If

If (*Condition*) IfClause [Else ElseClause]

The *If* statement is part of the If … Else structure, in which WILMA completes the instructions in the *IfClause* when the attached *If* condition is true, and skips instructions in the *ElseClause*. When the *If* condition is not true, WILMA skips the instructions in the *IfClause* and completes the instructions in the *ElseClause*. *If* clauses may be nested in *Else* clauses.

Argument	Description
Condition	Specifies a condition that, if true, causes WILMA to complete instructions in the IfClause; otherwise, WILMA completes instructions in the ElseClause. *Condition* can be either a non-zero value that evaluates to True (1) or a value that evaluates to False (0).
IfClause	Specifies an instruction or set of instructions that WILMA completes when the If Condition is true. Multiple statements in the IfClause must be enclosed in braces ({ }).
ElseClause	Specifies an instruction or set of instructions that WILMA completes when the If Condition is false. Multiple statements in the ElseClause must be enclosed in braces ({ }). The ElseClause is optional.

Examples

See examples for the Else statement.

See Also

For more information on the If … Else structure, see page 26 of this manual.

IfErr (Error Test)

IfErr (*Expression, TrueVal, FalseVal*)

The *IfErr* function tests whether a specified expression evaluates to an error, and returns *TrueVal* if it does evaluate to an error, or *FalseVal* if it doesn't.

down from there. Since HLookup only searches the top row for *Value* until it finds an equal or higher value, the values in the top row should be in ascending order.

Examples

	A	B	C	D
1	East	West	North	South
2	69.56	71.93	74.30	76.67
3	79.04	81.41	83.78	86.15
4	88.52	90.89	93.26	95.63
5	98.00	100.37	102.74	105.11
6	107.48	109.85	112.22	114.59
7	116.96	119.33	121.70	124.07
8	126.44	128.81	131.18	133.55

HLookup ("North", A1:D8, 4); displays 102.74, the value in C5, which is the fourth cell below "North".

HLookup ("Central", A1:D8, 3); displays 95.63, the value 3 cells below the index row A1, in the **last** column of the range row (since "Central" is not in the index row).

HLookup (69.56, A2:D8, 5); displays 116.96, the value 5 cells below 69.56, which is in the index row A2.

Hour (Hour of the Day)

Hour (*DNumber*)

The *Hour* function returns an integer from 0 to 23 (inclusive), corresponding to the hour of the day for the date number you specify. 0 is midnight, 12 is noon, and 23 is 11:00 PM.

Argument	Description
DNumber	Specifies a valid date number (see page 54 of this manual for more on Time functions).

Examples

Hour (0.7) gives 16.

Hour (Time (8, 15, 30)) gives 8. The Time function here gives the fractional part of the date number that corresponds to the time 08:15:30 AM, which is then used as the argument *DNumber* for the Hour function.

HLookup

HLookup (*Value, Range, Offset*)

The *HLookup* command searches from left to right across the first row in the range specified by *Range* until it finds the specified *Value*. *HLookup* then moves down in that column *Offset* cells and returns the cell at that location.

HLookup returns the cell reference (for example D15). ParaSheet can use this reference to display the value in the referenced cell, which is the usual result of *HLookup*. However, you can also use *HLookup* as a range reference—for example, in the macro command Select.

Argument	Description
Value	Specifies the text or numeric value to be searched for. Enter a text value if the top row contains text; a numeric value if the top row contains numeric values.
Range	A range specifying the index row (the top row of a lookup table) — that is, the row in which HLookup should search for *Value*. If *Range* is a range containing more than one row, HLookup searches only the top row of the range, from the first cell to the last column in the range.
Offset	A positive integer representing the number of cells to move below the index row. When the *Value* is text, HLookup searches the top (index) row for an exact match. For example, if *Value* is "Total", HLookup will locate only "Total", not "total" or "TOTAL". If HLookup doesn't find an exact match, it searches to the last column in the range *Range* before moving down offset cells, then returns that cell's reference. When *Value* is numeric, HLookup searches until it finds either *Value* or a number larger than *Value*. If it finds the specified value, HLookup stops at that cell and moves down *Offset* cells. Otherwise, it searches until it locates a larger value, backs up one cell, and moves down *Offset* cells from that cell. If *Value* is larger than all the numbers in the first row of *Range*, HLookup searches to the last column in the range *Range* and moves

with zero). Otherwise, it searches until it locates a larger value, backs up one cell, and gives the count of the cells searched up to that cell. If *Value* is larger than all the numbers in the first row in *Range*, HIndex gives the count of **all** the cells in that row starting with 0, up to the last column in *Range*. Since HIndex only searches the top row for *Value* until it finds an equal or higher value, the values in the top row should be in ascending order.

Range — A range that specifies the index row (top row of a lookup table) — that is, the row in which HIndex should search for *Value*. If *Range* contains more than one row, HIndex will search only the top row of the range, beginning with the first cell, and ending with the cell in the last column of the range.

Examples

	A	B	C	D
1				
2	42	66	188	320
3	31	56	157	275
4	53	112	83	215
5	65	324	201	245
6	26	44	227	201
7	84	65	199	198
8	99	101	175	189

HIndex (188,A2:D8); gives 2, the count of cells searched in row 2 to locate the value 188.

HIndex (200,A2:D8); gives 2, the count of cells searched in row 2 **before** a value higher than 200 was located.

Argument	Description
Setting	Specifies whether ParaSheet is hidden from view. Acceptable entries are: "Yes" or 1 "No" or 0

Hide will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section page 18 of this manual.

Examples

Hide ("Yes"); hides ParaSheet.

Hide (0); displays ParaSheet.

HIndex

HIndex (*Value, Range*)

The *HIndex* command searches from left to right across the first row in the range specified by *Range* until it finds the specified *Value*, and returns the count of the cells searched. As it searches, *HIndex* counts cells starting at zero. For example, if two cells are searched to locate *Value*, HIndex returns 1.

Argument	Description
Value	The text or numeric value to be searched for. Use a text value if the top row contains text; use a numeric value if the top row contains numeric values. When *Value* is text, HIndex searches for an exact match. For example, if *Value* is "Total", HIndex will locate only "Total", not "total" or "TOTAL". If an exact text match isn't found, HIndex gives the count (beginning with zero) of **all** the cells in the first row of the range specified by *Range*. When *Value* is numeric, HIndex searches until it finds either *Value*, or a number larger than *Value*. If it finds *Value*, HIndex gives the count of the cells searched (beginning

HeadingsShow

HeadingsShow (*What, Setting*)

The *HeadingsShow* command specifies whether the row headings (1, 2, 3 …) and column headings (A, B, C …) show when the spreadsheet is displayed and printed. This command is equivalent to the *Show Column Heading/Hide Column Heading* and *Show Row Heading/Hide Row Heading* commands in the Layout submenu.

Argument	Description
What	Specifies which headings show. Acceptable entries are: "Row" or 1 "Col" or 2 "Both" or 3
Setting	Specifies whether column and row headings display and print. Acceptable entries are: "Yes" or 1 "No" or 0

Examples

HeadingsShow ("Row", "Yes"); causes row headings in the current spreadsheet to display (and print).

HeadingsShow (3, 1); causes row and column headings to display (and print).

HeadingsShow ("Both", 0); hides row and column headings in the current spreadsheet.

Hide

Hide (*Setting*)

The *Hide* command conceals ParaSheet and its windows from view in the Workspace Manager. This command is equivalent to the *Hide* command in the main menu.

Argument	Description
Reference	Specifies a cell, range of cells, named range, or object name or number. If the reference is a range of cells or named range, the top left cell will be selected and the rest of the range will be highlighted.

Examples

GoTo (P44); selects the cell at column P, row 44.

GoTo (MyNamedRange); selects the top left cell in the range named MyNamedRange.

GoTo ("GrayOval"); selects the object named GrayOval.

GridShow

GridShow (*Setting*)

The *GridShow* command specifies whether spreadsheet grid lines show when the spreadsheet is displayed and printed. This command is equivalent to the *Show Grid/Hide Grid* commands in the Layout submenu.

Argument	Description
Setting	Specifies whether grid lines in the spreadsheet window show. Acceptable entries are: "Yes" or 1 "No" or 0

Examples

GridShow ("Yes"); causes the grid lines in the current spreadsheet to display (and print).

GridShow (0); hides the grid lines in the current spreadsheet.

Finds the size of a spreadsheet window named Resources. If the window is less than 675 pixels (approximately 7 1/3 inches) wide or 421 pixels (approximately 4.5 inches) tall, this macro resizes it to 675 by 421 pixels (the default size).

Global

Global (*VariableList*)

The *Global* statement declares and optionally assigns values to one or more global variables. Global variables are accessible from all ParaSheet spreadsheets and macros. If you don't assign a value to a global variable, it is initialized to the integer value 0. A global variable exists until you delete it.

Argument	Description
VariableList	Specifies one or more variables. Multiple variables must be separated by commas.

Examples

Global (Q_1, Q_2, Q_3, Q_4); declares four variables using the specified names — values can be assigned to these variables later in the macro.

Global (ReportTitle := "Budget Report"); assigns a text string to the variable named ReportTitle.

Global (Results := Sum (Barrels)); assigns the sum of a named range called Barrels to the global variable Results. If the named range doesn't exist, a run-time error alert panel appears.

See Also

For more information on variables, see the "Using Variables" section on page 20 of this manual.

GoTo

GoTo ([*Reference*])

The *GoTo* command selects the specified cell, range of cells, or object. This command is equivalent to the *Go To* command in the Sheet menu. If you don't provide any arguments, the Go To panel appears.

GetWindowSize

GetWindowSize (*WhichCoordinate* [, *Filename*])

The *GetWindowSize* command returns the width or height of a spreadsheet window.

Returns	Description
A positive integer (pixels)	Indicates the width or height of the window

Argument	Description
WhichCoordinate	Specifies whether ParaSheet measures the window horizontally or vertically. Acceptable entries are: "Width" or 1 "Height" or 2
Filename	Specifies the name of the spreadsheet. If the spreadsheet is in another folder, you must provide a path name. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename (including path name) must be surrounded by quotation marks. If you don't provide a filename, ParaSheet returns a measurement for the currently active spreadsheet.

Example

```
Local ( HowWide, HowTall );

HowWide := GetWindowSize ( "Width", "/Data/Spreadsheets/
Resources" );

HowTall := GetWindowSize ( "Height", "/Data/Spreadsheets/
Resources" );

If ( HowWide < 675 #or# HowTall < 421 )

WindowSize ( 675, 421);
```

Argument	Description
WhichCoordinate	Specifies whether ParaSheet measures horizontal or vertical distance. Acceptable entries are: "Horiz" or 1 "Vert" or 2
Filename	Specifies the name of the spreadsheet. If the spreadsheet is in another folder, you must provide a path name. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename (including path name) must be surrounded by quotation marks. If you don't provide a filename, ParaSheet returns the position of the currently active spreadsheet.

Example

```
Local ( WhereHoriz, WhereVert );

WhereHoriz := GetWindowPosition ( "Horiz", "/Data/Spreadsheets/
Resources" );

WhereVert := GetWindowPosition ( "Vert", "/Data/Spreadsheets/
Resources" );

If ( WhereHoriz < 184 #or# WhereVert < 138)

WindowMove ( 184, 138, "Abs" );
```

Finds the coordinates of a spreadsheet named Resources. If Resources is less than 184 pixels (2 inches) to the right of or 138 pixels (1.5 inches) below the top left corner of the screen, this macro moves it.

```
{
FormatDebt ( );
DebtAction ( );
}
```

This user-defined function notes its cell location in the spreadsheet and stores the column and row coordinates in variables named SourceCol and SourceRow. It keeps track of the value in a range named TargetCell (a named range consisting of a single cell, located one cell above) and compares it to the value in a range named BudgetLimit (another named range). If the value in TargetCell exceeds the value in BudgetLimit, this macro calls two UDFs: one for custom formatting and one to analyze and archive the results.

GetUDFRow

GetUDFRow ( )

The *GetUDFRow* command returns a positive integer representing the vertical coordinate of the cell from which a user-defined function was running (1 represents row 1, 2 represents row 2, and so on).

There are no arguments for *GetUDFRow*, but parentheses must be included.

Example

See the example for GetUDFCol, above.

GetWindowPosition

GetWindowPosition (*WhichCoordinate* [, *Filename*])

The *GetWindowPosition* command returns the location of a spreadsheet window relative to the top left corner of the screen.

Returns	Description
A positive integer (pixels)	Represents the distance from the top or left corner of the screen

Selection	Specifies a range of rows or columns. If you don't provide a selection, ParaSheet returns the size of the rows or columns in the current selection.

Example

```
Local ( SizeCols, SizeRows );
SizeCols := GetSize ( "Cols", J17 );
SizeRows := GetSize ( "Rows", J17 );
If ( SizeCols < 78 #or# SizeRows < 16)
    {
    SizeSet ( "Row", 16, J17 );
    SizeSet ( "Col", 78, J17 );
    }
```

Finds the height and width of cell J17. If the row height is less than 16 pixels (the default) or the column width is less than 78 pixels (the default), this macro resizes column J and row 17 to the default size.

GetUDFCol

GetUDFCol ( )

The *GetUDFCol* command returns a positive integer representing the horizontal coordinate of the cell from which a user-defined function was running (1 represents column A, 2 represents B, and so on).

There are no arguments for *GetUDFCol*, but parentheses must be included.

Example

```
Local ( SourceCol, SourceRow, TargetCell );
SourceCol := GetUDFCol ( );
SourceRow := GetUDFRow ( );
TargetCell := Select ( SourceCol, SourceRow – 1 );
If ( TargetCell > BudgetLimit )
```

A positive integer	For an object, if the current selection is an unnamed object

There are no arguments for *GetSelection*, but parentheses must be included.

Example

```
Local ( WhatSelection );
Select ( K2:L26 );
WhatSelection := GetSelection ( );
If ( WhatSelection = "Parrots" )
    {
    Cut ( Parrots );
    Paste ( TropicalBirds );
    }
```

Finds the coordinates of the named range Parrots. If the coordinates match the specified range, this macro cuts the contents of Parrots and pastes them in the range named TropicalBirds.

GetSize

GetSize (*WhichAttribute* [, *Selection*])

The *GetSize* command returns the height of a row, or width of a column.

Returns	Description
A number (pixels)	Represents the row height or column width. If you specify multiple rows or columns in the selection, ParaSheet returns the size of the top left cell.

Argument	Description
WhichAttribute	Specifies whether ParaSheet returns the size of rows or columns. Acceptable entries are: "Rows" or 1 "Cols" or 2

GetRowNum

GetRowNum ([*Selection*])

The *GetRowNum* command returns the index number of the top row in a range of cells or under an object.

Returns	Description
An integer	Represents the index number of the top row in the specified range or under an object. The index number for row 1 is 1, row 2 is 2, and so on.

Argument	Description
Selection	Specifies any range of cells, named range, or object name or number. If you don't provide a selection, ParaSheet gets the row number of the current selection.

Examples

GetRowNum (B3:D14) gives 3, the top row in the range B3:D14.

GetRowNum (Sales) gives 5 when Sales is the name of the range A5:E15.

GetSelection

GetSelection ( )

The *GetSelection* command returns the range name, object name or object number of a selection.

Returns	Description
A range name	For a cell or range of cells, if the current selection is a named range
–1	For a cell or range of cells, if the current selection is an unnamed range, multiple named ranges, or a named range and additional cells
An object name	For an object, if the current selection is a named object

Example

```
Local ( Start );
Start := GetPieStartAngle ( "Wildlife" );
If ( Start <= 180 )
        PieStartAngleSet ( 270, "Wildlife" );
```

Finds the starting angle of a pie chart named Wildlife. If the starting angle is less than 180 (on the right side of the pie), this macro resets the starting angle at 270 degrees (9 o'clock).

GetRange

GetRange ([*Selection* **])**

The *GetRange* command returns the selected range's name if one exists.

Returns	Description
A range reference	Indicates the name of the range if one exists
–1	Indicates the range is unnamed

Argument	Description
Selection	Specifies a range of rows or columns. If you don't provide a selection, ParaSheet returns the range name or reference for the current selection.

Example

```
Local ( WhatRange );
WhatRange := GetRange ( K2:L16 );
If ( WhatRange = "Parrots" )
NamedRngEdit ( "Parrots", "TropicalBirds", K2:L26 );
```

Finds the range name for cells K2:L6. If the name is Parrots, this macro changes it to TropicalBirds, and increases the size of the range by 10 cells in each column.

Example

```
Local ( Labels, Values );
Labels := GetPieMarks ( "Label", 3);
If ( Labels = 0 )
    PieMarksSet ( "Yes", "Yes", "No", "Yes" );
Else
    {
    Values := GetPieMarks ( "Value", 3 )
    If ( Values = 1 )
        PieMarksSet ( "Yes", "Yes", "No", "Yes" );
    }
```

Checks whether the pie chart labels are showing and, if not, makes sure they show. Then, this macro sets only categories and percentages, but not values, to display. If labels are showing, this macro still makes sure that only categories and percentages display.

GetPieStartAngle

GetPieStartAngle ([*PieChartID*])

The *GetPieStartAngle* command returns the starting angle of the first slice (the first plotted value) of a pie chart. This command corresponds to the Starting Angle setting in the View Options subpanel in the Chart Inspector.

Returns	Description
An integer	Represents the angle (measured in degrees) at which the pie chart starts: an integer between 0 and 359, inclusive (0 is the top of the circle; 180 is the bottom).

Argument	Description
PieChartID	Specifies the name or number of the pie chart. If you don't provide a *PieChartID*, ParaSheet returns the starting angle of the currently selected pie chart.

Returns	Description
A positive integer	Indicates the page number

Argument	Description
Selection	Specifies the cell, range of cells, named range, or object name or number. If you don't provide a selection, ParaSheet returns the page number of the current selection.

Examples

GetPageNum (H1:J8) gives 2 if cell H1 is located on page 2.

GetPageNum ( ) gives 12 when it's typed in a cell located on page 12.

GetPieMarks

GetPieMarks (*WhichAttribute* [, *PieChartID*])

The *GetPieMarks* command returns which labels mark each slice of a pie chart. This command corresponds to settings in the Pie Labels area of the Marking submenu in the Chart Inspector.

Argument	Description
1 (or logical True)	If the specified attribute is visible
0 (or logical False)	If the specified attribute is not visible

Argument	Description
WhichAttribute	Specifies the attribute to be checked. Acceptable entries are: "Label" or 1 "Percent" or 2 "Value" or 3 "Category" or 4
PieChartID	Specifies the name or number of the pie chart. If you don't provide a *PieChartID*, ParaSheet returns label information for the currently selected pie chart.

GetObjSize

GetObjSize (*WhichCoordinate* [, *ObjectID*])

The *GetObjSize* command returns the width or height of an object.

Returns	Description
A number	Indicates the width or height (measured in pixels) of the object

Argument	Description
WhichCoordinate	Specifies whether ParaSheet measures the object horizontally or vertically. Acceptable entries are: "Width" or 1 "Height" or 2
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID*, ParaSheet returns a size measurement for the currently selected object.

Example

```
Local ( HowWide, HowTall );
HowWide := GetObjSize ( "Width", "BudgetChart" );
HowTall := GetObjSize ( "Height", "BudgetChart" );
If ( HowWide < 184 #or# HowTall < 92 )
ObjSize ( 184, 92 );
```

Finds the size of a chart named "BudgetChart". If the chart is less than two inches wide or one inch tall, this macro resizes it.

GetPageNum

GetPageNum ([*Selection*])

The *GetPageNum* command returns either the page number on which the top left corner of a selected range of cells or object is located, or the current page number.

Example

```
Local ( Number );
Number := GetObjNum ( "FreshWater" );
If ( Number > 5)
Cut ( "FreshWater" );
```

Finds the name of the object named FreshWater in the current spreadsheet. If the object number is greater than 5, this macro deletes it.

GetObjPosition

GetObjPosition (*WhichCoordinate* [, *ObjectID*])

The *GetObjPosition* command returns the spreadsheet location of an object relative to the top and left edges of the spreadsheet.

Returns	Description
A number	Represents the vertical or horizontal distance (measured in pixels) from the top left corner of the spreadsheet

Argument	Description
WhichCoordinate	Specifies whether ParaSheet measures horizontal or vertical distance. Acceptable entries are: "Horiz" or 1 "Vert" or 2
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID*, ParaSheet returns a distance measurement for the currently selected object.

	returns the name of the currently selected object.

Example

```
Local ( Name );
Name := GetObjName ( 2 );
If ( Name := –1 )
ObjInfo ( "DataChart" );
```

Finds the name of object number 2 (a chart) in the current spreadsheet. If object number 2 has no name, this macro names it DataChart.

GetObjNum

GetObjNum ([*ObjectID*])

The *GetObjNum* command returns the number of an object. This command corresponds to the Number setting in the Chart area of the Common subpanel in the Chart Inspector.

Returns	Description
A positive integer	Indicates the object number

Argument	Description
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID*, ParaSheet returns the number of the currently selected object.

Object numbers don't necessarily reflect the order in which objects have been created. If you delete an object and then create another object, the new object might have the same number as the deleted object.

	"YLabel" or 6 (except pie charts)
	"ZLabel" or 7 (for 3D charts only)
	"LegendBox" or 8
	"LegendEntry" or 9
	"Graph" or 10
	"PieLabel" or 11 (for pie charts only)
	"Side" or 12 (for 3D charts only, to count visible sides of graphed data)
ChartID	Specifies the name or number of the chart. If you don't provide a *ChartID*, ParaSheet returns the number of the specified chart type for the currently selected chart.

Example

```
Local ( NumXLabel, NumLegendEntry );
NumXLabel := GetNumParts ( "XLabel", "FreshWater" );
NumLegendEntry := GetNumParts ( 9, "FreshWater" );
Enter ( NumXLabel, G1 ); Enter ( NumLegendEntry, G2 );
```

Finds the number of x-axis labels and legend entries in a chart named FreshWater, and enters the results in cells G1 and G2.

GetObjName

GetObjName ([*ObjectID*])

The *GetObjName* command returns the name of an object. This command corresponds to the Name setting in the Chart area of the Common subpanel in the Chart Inspector.

Returns	**Description**
A name	Indicates the name of the object
–1	Indicates that the object is unnamed

Argument	**Description**
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID*, ParaSheet

GetNumPages

GetNumPages ( )

The *GetNumPages* command returns the total number of pages that will be printed in the active spreadsheet.

Returns	Description
A positive integer	Represents the number of pages

There are no arguments for *GetNumPages*, but parentheses must be included.

Example

GetNumPages ( ); returns 5 when the active spreadsheet has 5 printable pages.

GetNumParts

GetNumParts (*WhichPart* [, *ChartID*])

The *GetNumParts* command returns the total number of a specific type of chart part.

Returns	Description
A positive integer	Represents the number of chart parts of the specified type

Argument	Description
WhichPart	Specifies the type of chart part to be counted. Acceptable entries are: "BarTotals" or 1 (for stacked bar and horizontal stacked bar charts only) "BarValues" or 2 (for bar and horizontal bar charts only) "ChartText" or 3 "DataSeries" or 4 "XLabel" or 5 (except pie charts)

GetNumObjs

GetNumObjs ([*Filename*])

The *GetNumObjs* command returns the total number of objects in a spreadsheet.

Returns	Description
A positive integer	Represents the number of objects
0	If there are no objects in the specified spreadsheet

Argument	Description
Filename	Specifies the name of the spreadsheet. If the spreadsheet is in another folder, you must provide a path name. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename (including path name) must be surrounded by quotation marks. If you don't provide a filename, ParaSheet returns the number of objects in the currently active spreadsheet.

Example

```
Local ( TotalObjs );
TotalObjs := GetNumObjs ( );
Enter ( TotalObjs, A1 );
```

Finds the total number of objects in the current spreadsheet, assigns that number to a variable, and displays the results in cell A1.

GetInterp

GetInterp ([*ChartID*])

The *GetInterp* command returns whether interpolated values have been assigned to non-numeric cells in the data set being charted. This command corresponds to the *Interpolate Non-numeric Cells* check box in the View Options subpanel in the Chart Inspector. It is not applicable to high-low or pie charts.

Returns	Description
1 (or logical True)	If non-numeric cells have been interpolated
0 (or logical False)	If non-numeric cells have not been interpolated

Argument	Description
ChartID	Specifies the name or number of a chart. If you don't specify a *ChartID*, ParaSheet returns interpolation settings for the currently selected chart.

Example

```
Local ( Interp );
Interp := GetInterp ( "OilConsumption" );
If ( Interp = 0 )
InterpSet ( "Yes", "OilConsumption" );
```

Checks whether non-numeric values are interpolated in the chart named OilConsumption and, if necessary, enables interpolation.

A positive integer	Represents the point size of the typeface
–1	Depending on what you specify for *WhichAttribute*, indicates that the selection has different fonts, typefaces, or sizes

Argument	**Description**
WhichAttribute	Specifies what font information ParaSheet returns. Acceptable entries are: "Name" or 1 for the font family name "Type" or 2 for the typeface name "Size" or 3 for the point size
Selection	A cell, range of cells, or named range. If you don't provide a selection, ParaSheet returns font information for the currently selected data.

To get the font of the contents of a chart text object, you must first select the chart text object using the ChartSelect command. Then, use the *TextSelect* command to select its contents. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object or label are already selected.)

Example

```
Local ( Font );
Select ( DataRange );
Font := GetFont ( "Name" );
If ( Font <> "Helvetica" )
FontSet ( "Helvetica", "Italic", 14, DataRange );
```

Selects the named range called DataRange, then finds the name of the font in which the range is displayed. If it is not displayed in Helvetica, this macro resets the font to 14-point Helvetica italic.

hhmm	For the hhmm format
m/d/yy h:mm	For the m/d/yy h:mm format
A name	For a custom format
–1	For a selection with multiple formats

Argument	**Description**
Selection	A cell, range of cells, or named range. If you don't provide a selection, ParaSheet returns predefined or custom format identification for the currently selected data.

To get the format of an axis label, you must first select the label using the ChartSelect command. (Using *ChartSelect* is unnecessary if the contents of the label are already selected.)

Example

```
Local ( Format );
Select ( DataRange );
Format := GetFmtType ( );
If ( Format <> "Scientific" )
FmtTypeSet ( "Scientific" );
```

Selects the named range called DataRange, then finds the name of the format in which the range is displayed. If it is not displayed in Scientific format, this macro reformats DataRange in the Scientific format.

GetFont

GetFont (*WhichAttribute* [, *Selection*])

The *GetFont* command returns either the font family, typeface, or size of characters in a range of cells, chart text object, or of a chart label. This command corresponds to settings in the Font panel.

Returns	**Description**
A font family name	Indicates the family name of the font
A typeface name	Indicates the name of the typeface

Finds the fill shade of a chart named BudgetChart, then checks how dark it is. If it is darker than the shade of gray represented by 0.9, this macro changes the fill shade to the light gray value 0.9.

GetFmtType

GetFmtType ([*Selection*])

The *GetFmtType* command returns the format applied to the selected data. (These formats correspond to predefined and custom formats in the ParaSheet Format Inspector.)

Returns	Description
General	For the General format
Currency 1	For the Currency 1 format
Currency 2	For the Currency 2 format
Currency 3	For the Currency 3 format
Currency 4	For the Currency 4 format
Scientific	For the Scientific format
Fixed Decimal 1	For the Fixed Decimal 1 format
Fixed Decimal 2	For the Fixed Decimal 2 format
Percent 1	For the Percent 1 format
Percent 2	For the Percent 2 format
Integer 1	For the Integer 1 format
Integer 2	For the Integer 2 format
m/d/yy	For the m/d/yy format
d-mmm-yy	For the d-mmm-yy format
d-mmm	For the d-mmm format
mmm-yy	For the mmm-yy format
ddmmmyy	For the ddmmmyy format
h:mm AM/PM	For the h:mm AM/PM format
h:mm:ss AM/PM	For the h:mm:ss AM/PM format
hh:mm	For the hh:mm format
hh:mm:ss	For the hh:mm:ss format

GetDocName

GetDocName ( )

The *GetDocName* command returns the name of the active spreadsheet.

There are no arguments for *GetDocName*, but parentheses must be included.

Example

GetDocName ( ); returns "QuarterlyReport.parasheet" when the active spreadsheet is named QuarterlyReport.

GetFillShade

GetFillShade ([*ObjectID*])

The *GetFillShade* command returns the gray-scale shade of the area within the borders of an object or chart part. This command corresponds to settings in the *Fill* color well in the Color subpanel in the Chart Inspector.

Returns	Description
A positive number	Represents the shade setting (0.0 is black and 1.0 is white)
–1	Indicates that the fill shade is set to None
–2	Indicates that multiple objects are selected

Argument	Description
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID*, ParaSheet returns a shade setting for the currently selected object.

Example

```
Local ( Shade );
Shade := GetFillShade ( "BudgetChart" );
If ( Shade < 0.9 )
FillShade ( 0.9, "BudgetChart" );
```

Example

```
Local ( Name );
Name := GetChartAttribs ( "Name" );
Name := Name & "1990";
ChartAttribsSet ( Name, "Cols", "ShowLegend", "Freeze" );
```

Finds the name of a selected chart, assigns it to a variable, and appends 1990 to the name. Then, this macro reassigns the new name to the chart, and also resets the source of chart labels, whether the legend shows, and freezes the graph size.

GetColNum

GetColNum ([*Selection*])

The *GetColNum* command returns the index number of the column furthest left in a range of cells, or under an object.

Returns	Description
An integer	Represents the index number of the column furthest left in the specified range (or under the left side of an object). The index number for column A is 1, B is 2, and so on.

Argument	Description
Selection	Specifies any range of cells or named range, or an object name or number. If you don't provide a selection, ParaSheet gets the column number of the current selection.

Examples

GetColNum (B3:L15); gives 2, the index number of column B.

GetColNum (L15:B3); gives 2, the index number of column B, even though the range is given from right to left.

GetColNum (D30:M55); gives 4, the index number of column D.

GetChartAttribs

GetChartAttribs (*WhichAttribute* [, *ChartID*])

The *GetChartAttribs* command returns either a name, source of chart labels, presence of legend, or whether the graph size is frozen for the specified chart. This command corresponds to settings in the Common subpanel of the Chart Inspector.

Returns	Description
A name	Indicates the name of the chart
–1	Indicates the chart is unnamed
1	For labels, if the source of chart labels is rows
2	For labels, if the source of chart labels is columns
0	For the legend, if it is not visible
1	For the legend, if it is visible
0	For the graph, if graph size is not frozen
1	For the graph, if graph size is frozen

Argument	Description
WhichAttribute	Specifies the particular attribute to check. Acceptable entries are: "Name" or 1 for the chart name "Labels" or 2 for the chart labels "Legend" or 3 for the legend "Graph" or 4 for whether the graph is frozen
ChartID	Specifies the name or number of the chart. If you don't provide a *ChartID*, ParaSheet returns a setting for the specified attribute of the currently selected chart.

Finds the border shade of a chart named BudgetChart, then checks how dark it is. If it is darker than the shade of gray represented by 0.9, this macro changes the border shade to the light gray value 0.9.

GetBorderWidth

GetBorderWidth ([*ObjectID*])

The *GetBorderWidth* command returns the border thickness of an object. This command corresponds to the numeric setting in the *Border Weight* area of the Color subpanel in the Chart Inspector.

Returns	Description
A positive number	Represents the width setting (0.0 is the thinnest setting and 10.0 is the thickest)
–2	Indicates that multiple objects are selected

Argument	Description
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID,* ParaSheet returns a width setting for the currently selected object.

Example

```
Local ( Width );
Width := GetBorderWidth ( "BudgetChart" );
If ( Width <> 3.75 )
BorderWidth ( 3.75, "BudgetChart" );
```

Finds the border width of a chart named BudgetChart, then checks whether the width matches a specific setting (3.75). If it does not, this macro changes the border width to 3.75.

Example

```
Local ( Scale );
Scale := GetAxisYScale ( "OilConsumption" );
If ( Scale <> 3 )
AxisScale ( 3 );
```

Finds the value at which the y-axis of a chart named OilConsumption is scaled. If the value is not 3 (percent scaling), this macro changes the scale setting to percent scaling.

GetBorderShade

GetBorderShade ([*ObjectID*])

The *GetBorderShade* command returns the gray-scale border shade of an object. This command corresponds to settings in the *Border* color well in the Color subpanel in the Chart Inspector.

Returns	Description
A positive number	Represents the shade setting (0.0 is black and 1.0 is white)
–1	Indicates that the border shade is set to None
–2	Indicates that multiple objects are selected

Argument	Description
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID*, ParaSheet returns a shade setting for the currently selected object.

Example

```
Local ( Shade );
Shade := GetBorderShade ( "BudgetChart" );
If ( Shade < 0.9 )
BorderShade ( 0.9, "BudgetChart" );
```

Example

```
Local ( Scale );
Scale := GetAxisXScale ( "OilConsumption" );
If ( Scale <> 2 )
AxisScale ( 1 , 2 );
```

Finds the value at which the x-axis of a chart named OilConsumption is scaled. If the value is not 2 (logarithmic), this macro changes the scale setting to logarithmic.

GetAxisYScale

GetAxisYScale ([*ChartID*])

The *GetAxisYScale* command returns the type of scaling for the y-axis of a chart. This command corresponds to settings in the Axis Scaling area of the View Options subpanel in the Chart Inspector. It is not applicable to pie charts.

Returns	**Description**
1	For an axis with linear scaling
2	For an axis with logarithmic scaling
3	For an axis with percent scaling

Argument	**Description**
ChartID	Specifies the name or number of a chart. If you don't provide a ***ChartID***, ParaSheet returns a scaling setting for the currently selected chart.

```
If (Value <> End )
Changed := True;
If ( Changed = True )
    {
    Number := GetAxisTicksValues ( "Number", "OilConsumption" );
    AxisTicksCalcMan ( Number, Start–5, End+5 );
    }
```

Disables automatic tick mark calculation, then finds the minimum and maximum data values in a named range called OilData and compares them with the current settings of a chart plotted from the same range. If the tick marks are set at inappropriate values for the data, this macro adjusts them.

GetAxisXScale

GetAxisXScale ([*ChartID*])

The *GetAxisXScale* command returns the type of scaling for the x-axis of a scatter chart. This command corresponds to settings in the Axis Scaling area of the View Options subpanel in the Chart Inspector.

Returns	Description
1	For an axis with linear scaling
2	For an axis with logarithmic scaling
3	For an axis with percent scaling

Argument	Description
ChartID	Specifies the name or number of a scatter chart. If you don't provide a *ChartID*, ParaSheet returns a scaling setting for the currently selected scatter chart.

Returns	Description
A number	Indicates the specified tick marks setting

Argument	Description
WhichSetting	Specifies the tick marks setting to check. For scatter charts, you may also use the last three settings. Acceptable entries are: "Num" or 1 (for number of tick marks on an axis) "Start" or 2 (for the starting value of the tick marks) "End" or 3 (for the ending value of the tick marks) "XNum" or 4 (for the number of tick marks on the x-axis of a scatter chart only) "XStart" or 5 (for the starting value of tick marks on the x-axis of a scatter chart only) "XEnd" or 6 (for the ending value of tick marks on the x-axis of a scatter chart only)
ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet returns a tick marks setting for the currently selected chart.

Example

```
Local ( Number, Start , End , Value, Changed := False );
Select ( "OilConsumption" );
AxisTicksCalcAuto ( "No" );
Start := GetAxisTicksValues ( "Start", "OilConsumption" );
Value := Min ( OilData );
If ( Value <> Start )
Changed := True;
End := GetAxisTicksValues ( "End", "OilConsumption" );
Value  := Max ( OilData );
```

Returns	Description
1 (or logical True)	If tick marks are showing
0 (or logical False)	If tick marks are not showing

Argument	Description
WhichAxis	Specifies the axis to check. Acceptable entries are: "XAxis" or 1 "YAxis" or 2 "ZAxis" or 3 (for 3D charts only)
ChartID	Specifies the name or number of a chart. If you don't specify a *ChartID*, ParaSheet returns a tick marks setting for the currently selected chart.

Example

```
Local ( XTick, YTick, ZTick);
Select ( "OilConsumption" );
XTick := GetAxisTicksShow ( "XAxis" );
YTick := GetAxisTicksShow ( "YAxis" );
ZTick := GetAxisTicksShow ( "ZAxis" );
If ( XTick = 0 #or# YTick = 0 #or# ZTick = 0 )
AxisTicksShow ( "Yes", "Yes", "Yes" );
```

Selects a chart named OilConsumption and checks whether tick marks for the x-axis, y-axis, or z-axis are displayed. If tick marks for any of the axes are hidden, this macro changes their settings so they all display.

GetAxisTicksValues

GetAxisTicksValues (*WhichSetting* [, *ChartID*])

The *GetAxisTicksValues* command returns information describing the settings of tick marks on the y-axis (or the x-axis for scatter charts). This command corresponds to settings in the Ticks area of the Marking subpanel in the Chart Inspector. It is not applicable to pie charts.

Returns	Description
1 (or logical True)	If tick marks are calculated automatically
0 (or logical False)	If tick marks are calculated manually

Argument	Description
WhichAxis	Specifies the axis to check. Acceptable entries are: "XAxis" or 1 (for scatter charts only) "YAxis" or 2
ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet returns a tick marks calculation setting for the currently selected chart.

Example

```
Local ( Auto );
Auto := GetAxisTicksCalc ( "YAxis", "OilConsumption" );
If ( Auto = 1 )
    {
    AxisTicksCalcAuto ( "No" );
    AxisTicksCalcMan ( 15, 0, 100 );
    }
```

Checks whether tick marks are calculated automatically or manually for a chart named OilConsumption. If they are calculated automatically, the setting is disabled so they can be set at custom values as shown above.

GetAxisTicksShow

GetAxisTicksShow (*WhichAxis* [, *ChartID*])

The *GetAxisTicksShow* command returns whether tick marks for a given chart axis are showing. This command corresponds to the Ticks settings in the Show area of the Marking subpanel in the Chart Inspector.

Returns	Description
1 (or logical True)	If the axis label is displayed
0 (or logical False)	If the axis label is not displayed

Argument	Description
WhichAxis	Specifies the axis labels to check. Acceptable entries are: "XAxis" or 1 "YAxis" or 2 "ZAxis" or 3 (for 3D charts only)
ChartID	Specifies the name or number of a chart. If you don't specify a *ChartID*, ParaSheet returns a label setting for the currently selected chart.

Example

```
Local ( XLabel, YLabel, ZLabel );
Select ( "OilConsumption" );
XLabel := GetAxisLabelsShow ( "XAxis" );
YLabel := GetAxisLabelsShow ( "YAxis" );
ZLabel := GetAxisLabelsShow ( "ZAxis" );
If ( XLabel = 0 #or# YLabel = 0 #or# ZLabel = 0 )
AxisLabelsShow ( "Yes", "Yes", "Yes" );
```

Selects a 3D chart named OilConsumption and checks whether labels for the x-axis, y-axis, or z-axis are displayed. If labels for any of the axes are hidden, this macro changes their settings so they all display.

GetAxisTicksCalc

GetAxisTicksCalc (*WhichAxis* [, *ChartID*])

The *GetAxisTicksCalc* command returns whether tick marks are calculated automatically or from user-defined values. This command corresponds to the Automatic Calculation check box in the Ticks area of the Marking subpanel in the Chart Inspector.

	1	If the label is to the left of the x-axis (for a horizontal bar or horizontal stacked bar chart)
	2	If the label is to the right of the x-axis (for a horizontal bar or horizontal stacked bar chart)

Argument	**Description**
WhichAxis	Specifies the axis to check. Acceptable entries are: "XAxis" or 1 (except pie and 3D charts) "YAxis" or 2 (except pie charts)
ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet returns a label setting for the currently selected chart.

Example

```
Local ( Display1 );
Select ( "OilConsumption" );
Display1 := GetAxisLabels ( "YAxis" );
If ( Display1 = 2 )
Display1 := "Above";
AxisLabelsSet ( Display1 );
```

Selects a chart named OilConsumption (a horizontal bar chart) and checks whether *y*-axis labels display above or below the graph. If they are set to display below the graph, this macro changes them to display above the graph.

GetAxisLabelsShow

GetAxisLabelsShow (*WhichAxis* [, *ChartID*])

The *GetAxisLabelsShow* command returns whether labels for a given chart axis are displayed. This command corresponds to the Labels settings in the Show area of the Marking subpanel in the Chart Inspector. It is not applicable to pie charts.

Example

```
Local ( XGrid, YGrid, ZGrid );
Select ( "OilConsumption" );
XGrid := GetAxisGridShow ( "XAxis" );
YGrid := GetAxisGridShow ( "YAxis" );
ZGrid := GetAxisGridShow ( "ZAxis" );
If ( XGrid = 0 #or# YGrid = 0 #or# ZGrid = 0 )
AxisGridShow ( "Yes", "Yes", "Yes" );
```

Selects the chart named OilConsumption and checks whether the x-axis, y-axis, or z-axis grid lines are displayed. If grid lines for any of the axes are hidden, this macro changes their settings so they all display.

GetAxisLabels

GetAxisLabels (*WhichAxis* [, *ChartID*])

The *GetAxisLabels* command returns the location of the labels for the given axis of a chart. This command corresponds to settings in the Axis Labels area of the Marking subpanel in the Chart Inspector. It is not applicable to pie charts.

Which Axis	Returns	Description
LabelY	1	If the label is to the left of the y-axis
	2	If the label is to the right of the y-axis
	1	If the label is above the y-axis (for a horizontal bar or horizontal stacked bar chart)
	2	If the label is below the y-axis (for a horizontal bar or horizontal stacked bar chart)
LabelX	1	If the label is above the x-axis
	2	If the label is below the x-axis

Examples

```
Local ( Depth1, Depth2 );
Depth1 := Get3DViewDepth ( "XAxis", "SalesChart" );
Depth2 := Get3DViewDepth ( "ZAxis", "SalesChart" );
3DViewDepthSet ( Depth1 + 5, Depth2 + 5 );
```

Finds, then increases, the thickness and width settings of bars in a 3D bar chart.

GetAxisGridShow

GetAxisGridShow (*WhichAxis* [, *ChartID*])

The *GetAxisGridShow* command returns whether grid lines for a given axis are displayed. This command corresponds to *Grid* settings in the Show area of the Marking subpanel in the Chart Inspector. It is not applicable to pie charts.

Returns	**Description**
1 (or logical True)	If the specified grid is displayed
0 (or logical False)	If the specified grid is not displayed

Argument	**Description**
WhichAxis	Specifies the axis to check. Acceptable entries are: "XAxis" or 1 "YAxis" or 2 "ZAxis" or 3 (for 3D charts only)
ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet returns a grid line setting for the currently selected chart.

ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet returns an angle setting for the currently selected chart.

Examples

```
Local ( RotAngle, ElevAngle, PerspAngle );
RotAngle := Get3DViewAngle ( "Rotation", "SalesChart" );
ElevAngle := Get3DViewAngle ( "Elevation", "SalesChart" );
PerspAngle := Get3DViewAngle ( "Perspective", "SalesChart" );
3DViewAngleSet ( RotAngle, ElevAngle + 5, PerspAngle, "SalesChart" );
```

Finds the rotation, elevation, and perspective angles for a 3D chart named SalesChart, assigns each value to a variable, then resets the angles for the chart.

Get3DViewDepth

Get3DViewDepth (*WhichAxis* [, *ChartID*])

The *Get3DViewDepth* command returns the viewing depth of a specified axis for a 3D chart. This command corresponds to settings in the Depth area of the View Options subpanel in the Chart Inspector. *Get3DViewDepth* is not applicable to wireframe charts.

Returns	**Description**
A number (percentage)	Indicates the specified viewing depth setting

Argument	**Description**
WhichAxis	Specifies the axis. Acceptable entries are: "ZAxis" or 1 "XAxis" or 2 (for 3D bar charts only)
ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet returns a viewing depth setting for the currently selected chart.

	"Lines" or 2
	For high-low charts, acceptable entries are:
	"Stock" or 1
	"Hash" or 2
ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet returns display settings for the currently selected chart.

Examples

```
Local ( TotalsDisp );
TotalsDisp := Get2DView ( "BarTotals", "OilConsumption" );
If ( TotalsDisp = 0 )
2DViewSet ( "Yes" );
```

Checks whether bar totals are displayed in a chart named OilConsumption, and if they aren't, displays them.

Get3DViewAngle

Get3DViewAngle (*WhichAngle* [, *ChartID*])

The *Get3DViewAngle* command returns a setting for one of the viewing angles, or the apparent viewing distance for a 3D chart. This command corresponds to settings in the Angles area of the View Options subpanel in the Chart Inspector.

Returns	**Description**
A number (degrees)	Represents the specified angle setting
Argument	**Description**
WhichAngle	Specifies the angle to check.
	Acceptable entries are:
	"Rotation" or 1
	"Elevation" or 2
	"Perspective" or 3

Argument	Description
Pmt	A number specifying the amount of each payment (or deposit into an investment).
Rate	A number specifying the periodic interest rate.
NumPmts	A positive integer specifying the number of payment periods.

Example

FV (250, .11/12,12 * 3) returns $10605.78. This is the future value of 36 payments of $250, earning 11% interest annually (every 12 periods). The actual amount of money invested is $9000 (36 * $250).

Get2DView

Get2DView (*Display* [, *ChartID*])

The *Get2DView* command returns how chart data is displayed, depending on the type of chart. This command corresponds to check boxes in the Show area of the View Options subpanel in the Chart Inspector. It is not applicable to pie, area, or 3D charts.

Returns	Description
1 (or logical True)	If the specified item is displayed
0 (or logical False)	If the specified item is not displayed

Argument	Description
Display	For bar and horizontal bar charts, acceptable entries are: "Overlap" or 1 "BarValues" or 2 For stacked bar and horizontal stacked bar charts, acceptable entries are: "BarTotals" or 1 For line and scatter charts, acceptable entries are: "Points" or 1

Example

Formats ( ); brings up the Format Inspector. Note that this is not a modal panel (as opposed to Incremental Fill, for example) so the macro will not pause in order for you to choose a format.

Frac (Fractional Portion)

Frac (*Number*)

The *Frac* function returns the absolute (positive) value of the fractional portion of a specified number.

Argument	Description
Number	Specifies a numeric value. Acceptable entries are: A number (including a formula or constant) A cell containing a numeric value A range, the top left cell of which contains a numeric value

Examples

Frac (3.86) gives 0.86.

Frac (C1) gives 0.9565 when cell C1 contains the value –2.9565.

Frac (C2:D5) gives 0.107 when cell C2 contains the value 4.107.

FV (Future Value)

FV (*Pmt, Rate, NumPmts*)

The *FV* function returns the future (compounded) value of an investment based on the amount of interest earned during periodic payments. FV assumes payments are made at the end of each period. If payments are made at the beginning of each period, multiply the result by (1 + *rate*). ParaSheet calculates the future value according to the following formula:

$$[\ Pmt\ *\ (\ (\ (\ 1\ +\ Rate\ )^{\mathrm{NumPmts}}\ )\ -\ 1\ )\ ]\ /\ Rate$$

Instructions	Expressions to be repeated until *Condition* is no longer true.

Examples

For (Arg_1 := 1; Arg_1 <= 10; Arg_1 := Arg_1 + 1) initializes Arg1 to 1, loops nine additional times, then terminates when Arg1 equals 11.

```
local (Arg_1, Arg_2);
For(Arg_1:= 1; Arg_1<=10; Arg_1:=Arg_1+1)
    {
     For (Arg_2 := 1; Arg_2 <= 10; Arg_2 := Arg_2 + 1 )
         {
         nextcell(0, 1);
         enter ( num2c ( Arg_1 ) &" ,"& num2c (Arg_2));
         }
    nextcell(1, -10);
     }
```

This is an example of a nested For loop macro, that first declares two variables, Arg_1 and Arg_2, then walks through a 10 X 10 matrix of cells, entering the current row and column values (separated by a comma) from the For loop.

For (; ;) is an example of an infinite loop.

See Also

For more information on the For structure, refer to page 33 of this manual.

Formats

Formats ( )

The *Formats* command causes the Format Inspector to appear. This command is equivalent to the Format Inspector command in the Format menu.

There are no arguments for *Formats*, but parentheses must be included.

TextSelect command to select its contents. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object or label are already selected.)

Example

FontSet ("Times", "Italic", 14, CostComparison); sets the contents of a range named CostComparison to Times italic, size 14.

For

For ([*AssignList*]; [*Condition*]; [*ControlList*])

{ *Instructions* }

The *For* statement is a structure made of three expressions separated by semicolons. It causes WILMA to repeat an action a specified number of times. The first expression initializes one or more variables. The second consists of one or more test conditions for the variables. The third controls the value of variables while the For structure loops. Instructions in the For structure repeat until the condition being tested is no longer true.

Argument	Description
AssignList	One or more expressions (values or formulas), each assigned to a variable. You must first declare variables (using the Local or Global statements) before using For. Multiple expressions must be separated by commas.
Condition	The test case for *AssignList*. When the test fails, the macro skips instructions within the For structure and continues executing the instructions that follow. Acceptable entries are: Single or multiple expressions (values or formulas, each assigned to a variable) that evaluate to a single value. Multiple expressions must be separated by logical operators (for example, #and#, #or#, and #not#).
ControlList	One or more expressions that increment or decrement the variables in *AssignList*. Multiple expressions must be separated by commas.

	ParaSheet applies the format to the current selection.

Examples

FmtTypeSet ("d-mmm-yy", Days); changes the format of the cells in the named range called Days to the d-mmm-yy format.

FmtTypeSet ("Scientific", A3); changes the format of the data in cell A3 to the Scientific format.

FontSet

FontSet ([*Family, Typeface, Size* [, *Selection*]])

The *FontSet* command sets the font family, typeface, and size of characters in the specified range of cells or chart text object. These settings correspond to choices in the ParaSheet Font panel. If you don't provide any arguments, the Font panel appears.

Argument	Description
Family	Specifies the name of the font family, which must be installed in the Workspace Manager. The family name must be surrounded by quotation marks.
Typeface	Specifies a typeface in the chosen font family. The typeface must be surrounded by quotation marks. Acceptable entries are: "Plain" or 1 "Bold" or 2 "Italic" or 3 "BoldItalic" or 4
Size	Specifies a point size for the chosen typeface.
Selection	Specifies the cell, range of cells, or named range. If you don't provide a selection, ParaSheet sets the font for the current selection.

When setting the font in a chart text object or for a chart label, you must first select the chart text object using the *ChartSelect* command. Then, use the

Argument	Description
Type	Specifies the predefined format to be applied. Acceptable entries are: “General” “Currency 1” ($#,##0) “Currency 2” [Red]($#,##0) “Currency 3” ($#,##0.00) “Currency 4” [Red]($#,##0.00) “Scientific” (0.00E+0) “Fixed Decimal 1” (0.00) “Fixed Decimal 2” (#,##0.00) “Percent 1” (0%) “Percent 2” (0.00%) “Integer 1” (0) “Integer 2” (#,##0) “m/d/yy”“d-mmm-yy” “d-mmm” “mmm-yy” “ddmmmyy” “h:mm AM/PM” “h:mm:ss AM/PM” “hh:mm” “hh:mm:ss” “hhmm” “m/d/yy h:mm” A custom format name, surrounded by quotation marks. You must type format names exactly as they appear in the Format Inspector, including spaces.
Selection	Specifies a cell, range of cells, or a named range. If you don’t provide a selection,

Example

FmtCustomAdd ("PoundsNotation", "\£#", TotalCost); appends a user-defined format called PoundsNotation to the Format Inspector, and formats the named range called TotalCost in the new PoundsNotation format.

See Also

For more information on creating custom formats, refer to Chapter 9, page 118, in the *Tutorial and Documentation* manual.

FmtCustomDel

FmtCustomDel (*FormatName*)

The *FmtCustomDel* command removes a custom format from the list in the Format Inspector. (Predefined formats cannot be deleted.) Any spreadsheet data in the deleted custom format is reformatted using the predefined General format. This command is equivalent to selecting a format in the Format Inspector and clicking *Delete*.

Argument	Description
FormatName	Specifies the name of the custom format to be deleted. The format name must be surrounded by quotation marks.

Example

FmtCustomDel ("PoundsNotation"); deletes the custom format called PoundsNotation from the Format Inspector and reformats any data in that format to General.

FmtTypeSet

FmtTypeSet ([*Type* [, *Selection*]])

The *FmtTypeSet* command formats data in a range of cells or in chart labels using one of the ParaSheet predefined (or custom, if any exist) data formats. This command is equivalent to setting a format in the Format Inspector. If you don't provide any arguments, the Format Inspector appears.

Placeholder	Description
0	Required digit
#	Optional digit
.	Decimal
,	Thousands
%	Percent
E+, e+, E–, e–	Scientific notation
\"	Quotes (a double quote must be preceded by a backslash)
\\	Backslash (a backslash must be preceded by another backslash)
( ) + – $ / : (	These characters display in the cell according to their position in the pattern)
@	Text
d	Day (no leading zero)
dd	Day (leading zero)
ddd	Day (abbreviated)
dddd	Day (full name)
m	Month (no leading zero)
mm	Month (leading zero)
mmm	Month (abbreviated)
mmmm	Month (full name)
yy	Year (2 digit)
yyyy	Year (4 digit)
h	Hour (no leading zero)
hh	Hour (leading zero)
m	Minute (no leading zero)
mm	Minute (leading zero)
s	Second (no leading zero)
ss	Second (leading zero)
/	AM/PM separator
General	Includes the General format in a pattern

Argument	Description
Number	Specifies a numeric value. Acceptable entries are: A number (including a formula or constant) A cell containing a numeric value A range, the top left cell of which contains a numeric value

Examples

Floor (B5) gives 62 when cell B5 contains 62.17

Floor (D14) returns –54 when cell D14 contains –53.17.

FmtCustomAdd

FmtCustomAdd ([*Name, Pattern* [, *Selection*]])

The *FmtCustomAdd* command lets you create a custom formatting pattern for data. This command is equivalent to settings in the Format Inspector. If you don't provide any arguments, the Format Inspector appears.

Argument	Description
Name	Specifies a descriptive name (with a maximum of 31 characters) for the custom format. Format names are not case sensitive. The name must be surrounded by quotes.
Pattern	Specifies a pattern (with a maximum of 255 characters) for the custom format (see Table 2-1). Patterns must be created according to the rules described in Chapter 9 of the *Tutorial and Documentation* manual, and surrounded by quotes.
Selection	Specifies a cell, range of cells, or named range. If you don't provide a selection, ParaSheet applies the custom format to the current selection.

To avoid unexpected results later when setting custom formats, try not to repeat existing format names.

Example

Fill ("Right", A1:D10); copies the data in cells A1 through A10 into the range B1:D10.

FillShade

FillShade (*Shade* [, *ObjectID*])

The *FillShade* command sets a gray-scale shade for the area within the borders of the selected object. This command is equivalent to settings in the *Fill* color well of the Color subpanel in the Chart Inspector.

Argument	Description
Shade	A number between 0.0 and 1.0, indicating the gray-scale shade of the fill. 0.0 is black and 1.0 is white. –1 indicates no shade (transparent). The ParaSheet default is no shade.
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID*, ParaSheet applies the fill shade to the currently selected object.

When setting the fill shade of a chart text object, legend, graph, or 3D chart wall, you must first select the chart part using the *ChartSelect* command. (Using ChartSelect is unnecessary if the chart part is already selected.)

Example

FillShade (0.835, "BudgetChart"); gives the chart named BudgetChart a light gray background.

Floor (Floor Value)

Floor (*Number*)

The *Floor* function returns the largest integer less than or equal to a specified number.

False (Logical False)

False ( )

The *False* function returns the logical value False, which is equivalent to the numeric value 0.

There are no arguments for *False*, but parentheses must be included.

Examples

False ( ) = True ( ) gives the value False (0).

IfX (False ( ), B5, B6) gives the value in B6, since the condition being evaluated is not True.

Choose (0, True ( ), False ( )) gives False (0).

Fill

Fill (*Direction* [, *Selection*])

The *Fill* command copies data (including formulas) in a selected column or row of cells and pastes it into the remaining selected cells to the right if the selection is a column, or below if the selection is a row. This command is equivalent to the *Fill Right* and *Fill Down* commands in the Fill submenu.

Argument	Description
Direction	Specifies which cells are to be filled. Acceptable entries are: "Right" or 1 (when the selection is columns) "Down" or 2 (when the selection is rows)
Selection	Specifies the cell or range of cells to be filled. If you don't specify a selection, ParaSheet fills the current selection.

Fill will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

If you enter a number other than those listed above, ParaSheet returns !NUM.

!CIRC errors cannot be compared using the IfX function. !CIRC errors always propagate !CIRC when used in a formula.

Example

IfX (A1 = B1, "There is an error", "There is not an error") compares the contents of cells A1 and B1. Assume that cell A1 contains =Error(1). If cell B1 evaluates to !DIV, this formula returns the first text string. Otherwise, it returns the second text string.

Exp (Exponentiation of e)

Exp (*Number*)

The *Exp* function returns the number *e* (approximately 2.7182818) raised to the power of a specified number. *e* is the constant used as the base in a natural logarithm. To calculate powers of other bases, use the exponentiation operator (^). *Exp* is the inverse of the *Ln* function.

Argument	Description
Number	Specifies the power to which e is raised. Acceptable entries are: A numeric value (constant or formula) A cell that contains a numeric value A range, the top left cell of which contains a numeric value

Examples

Exp (0) gives 1.

Exp (1) gives 2.718282.

Exp (B1) gives 7.389056 when cell B1 contains 2.

Exp (Ln (Number)) gives Number.

There are no arguments for *Err*, but parentheses must be included.

Examples

IfX (B12 < 25, Err ( ), B12) gives !ERR if the value in cell B12 is less than 25. Otherwise it displays the value in cell B12. (See the description of the IfX function.)

IfX (C15 = 0, Err ( ), Mod (C15, D15)) gives !ERR if cell C15 contains zero. Otherwise, it displays the result of the calculation Mod (C15, D15).

Error (Error Code)

Error (*Number*)

The *Error* function returns the cell error code corresponding to a specified number. You can use the *Error* function with the *IfX* function to determine whether a cell contains a specific type of error, but only if the Error function is referred to indirectly by a cell reference (see example below).

Argument	Description
Number	Specifies the error code to return. Acceptable entries are: 1 for ERROR (1) (!DIV) 2 for ERROR (2) (!N/A. This is equivalent to the NA function.) 3 for ERROR (3) (!VALUE) 4 for ERROR (4) (!REF) 5 for ERROR (5) (!NUM) 6 for ERROR (6) (!NAME) 7 for ERROR (7) (!CIRC) 8 for ERROR (8) (!SYS) 9 for ERROR (9) (!EVAL) 10 for ERROR (10) (!ERR. This is equivalent to the ERR function.) 11 for ERROR (11) (!MACRO) 12 for ERROR (12) (!UDF)

	formula itself into a cell (as opposed to its result), the first character must be preceded by an equal (=), plus (+), or minus (–) sign and the entire formula must be surrounded by quotation marks (see example below).
Selection	Specifies the cells, columns, rows, or object into which data is entered. Acceptable entries are: a cell reference, range name, range reference or name. If you provide a range, data is entered into the top left cell of the range only. If you don't provide a selection, ParaSheet enters data into the current selection.

When entering text into a chart text object, you must first use *ChartSelect* to select the chart text object.

Enter will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section page 18 of this manual.

Examples

Enter ("Sales for 1991", B2); enters the specified text string into cell B2.

Enter (Sum (Sales1991), C25); sums the data represented by the named range Sales1991 and enters the results into cell C25.

Enter (b^2 – 4 * a * c / 2a, SquareRootValue); executes the formula b^2 – 4*a*c / 2a and places the results in the top left cell of the named range called SquareRootValue.

Enter (A1+1, A2); enters the computed result of A1 + 1 into cell A2.

Err (User Error)

Err ( )

The *Err* function returns the value !ERR, a ParaSheet error code. This function is equivalent to ERROR(10). You can use the *Err* function to identify data entry errors and other user errors in a spreadsheet, such as possible out-of-range values, as shown below. Any formula that refers to a cell containing a value of !ERR also gives the value !ERR. This prevents the occurrence of a chain of inaccurate calculations throughout a spreadsheet.

false. Multiple statements in the *ElseClause* must be enclosed in braces ({ }). The *ElseClause* is optional.

Example

```
If ( Sales > 1000 )
Alert ( "Alert", "Inventory is low." );
Else
    {
    Alert ( "Alert", "Enough inventory for order." );
    Enter ( "OK", Z1 );
    }
```

Checks if more than 1000 units have been sold and, if they have, displays an appropriate alert. If l000 or fewer units have been sold, this macro displays a different alert and enters text into cell Z1.

See Also

For more information on the If … Else structure, refer to page 26 of this manual.

Enter

Enter (*What* [, *Selection*])

Like the Return key, the *Enter* command enters data into a cell. *Enter* also inserts text into a chart text object, in which case it positions the I-beam immediately after the last character entered. When adding text to an existing chart text object, new text is entered at the current I-beam position. If text is selected, new text replaces the selected text. If the I-beam is not positioned, new text is entered at the beginning of the existing text.

Argument	Description
What	Specifies the data to be entered. Acceptable entries are: Text, numbers, or formulas. Text to be entered must be surrounded by quotation marks. If you want to enter a

DOW (Day of the Week)

DOW (*DNumber*)

The *DOW* function returns an integer from 1 to 7 (inclusive), corresponding to the day of the week for the date number you specify. 1 is Sunday, 7 is Saturday.

Argument	Description
DNumber	Specifies a valid date number (see page 54 of this manual for more on Time functions).

Example

DOW (32842) gives 4 (Wednesday).

Else

If (*Condition*)

IfClause

[Else

***ElseClause*]**

The *Else* statement is part of the If … Else structure, in which WILMA completes the instructions in the *ElseClause* instead of those in the *IfClause*. WILMA only completes the *ElseClause* if the condition attached to If is not true.

Argument	Description
Condition	Specifies a condition that, if true, causes WILMA to complete instructions in the *IfClause*; otherwise, WILMA completes instructions in the *ElseClause*. *Condition* can be either a non-zero value that evaluates to True (1) or a value that evaluates to False (0).
IfClause	Specifies an instruction or set of instructions that WILMA completes when the If Condition is true. Multiple statements in the I*fClause* must be enclosed in braces ({ }).
ElseClause	Specifies an instruction or set of instructions that WILMA completes if the If Condition is

Example

DB (2500, 15, 125%, 5) gives $147.10 — the amount to depreciate a $2500 investment the fifth year, using the declining balance method with a straight-line rate multiplier of 125% (1.25) and a depreciation term of 15 years.

Delete

Delete ([*ReplaceDirection* [, *Selection*]])

The *Delete* command removes cells, columns, or rows from the spreadsheet. Remaining data moves to the left or up in the spreadsheet, depending on what was deleted. Any absolute, relative, or mixed cell references are updated. Delete does not work with multiple selections. This command is equivalent to choices in the Delete panel (the *Delete* command in the Edit menu).

Argument	Description
ReplaceDirection	Specifies the direction in which remaining cells, rows, or columns will move after the deletion. If you don't specify a *ReplaceDirection*, a Delete panel appears requesting one. Acceptable entries are: "Left" or 1 "Up" or 2
Selection	Specifies the cells, columns, or rows to be deleted. Acceptable entries are: cell references, range names, column letters, row numbers. If you don't provide a selection, ParaSheet deletes the current selection.

Delete will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Examples

Delete ( ); brings up the Delete panel.

Delete ("Left", SalesTotal); deletes the named range SalesTotal and moves remaining cells to the left.

Day (Day of the Month)

Day (*DNumber*)

The *Day* function returns an integer from 1 to 31 (inclusive), corresponding to the day of the month for the date number you specify.

Argument	Description
DNumber	Specifies a valid date number (see page 54 of this manual for more on Time functions).

Examples

Day (31770) gives 25.

Day (Date (1997, 11, 14)) gives 14. The Date function here gives the date number for 11/14/1997, which is then used as the argument *DNumber* for the Day function.

DB (Declining Balance Depreciation)

DB (*Inv, Term, Mult, Period*)

The *DB* function returns the amount an original investment has decreased in value in a specified period. The formula ParaSheet uses to calculate the declining balance depreciation is:

$$Inv * Mult / Term * (1 - Mult / Term)^{(Period - 1)}$$

Argument	Description
Inv	A positive number specifying the initial investment.
Term	A positive number specifying the number of periods over which the initial investment decreases in value.
Mult	A number specifying the straight-line rate multiplier.
Period	A number specifying the period for which to calculate the depreciation.

outside the range of valid date numbers (0 to 65534.99999), ParaSheet returns !NUM.

Argument	Description
Year	Specifies the year. Acceptable entries are: An integer between 1904 and 2083 (inclusive). An integer between 4 to 183 (inclusive) is accepted as shorthand for years in the range 1904 to 2083.4 represents 1904, and 183 represents 2083.
Month	Specifies the month. Acceptable entries are: An integer between 1 and 12 (inclusive). 1 represents January, and 12 represents December. If you enter an integer greater than 12, the Date function continues counting months into the next year.
Day	Specifies the day of a month. Acceptable entries are: An integer between 1 and 31 (inclusive). If there is no such day in the given month, the Date function continues counting days into the next month.

Examples

Date (1992, 11, 14) gives a date number of 32460.

Date (92, 11, 14) also gives 32460.

Date (2004, 9, 12) and **Date (104, 9, 12)** both give 36780.

Date (91, 11, 31) and **Date (91, 12, 1)** both give 32111.

Date (1998, 9, 12) displays as 12-Sep-98 in a cell formatted for the date format D-Mon-YY.

Int (Date (90, 11, 14) – Date (90, 9, 30)) gives 45 — the number of days between 9/30/90 and 11/14/90. The date number corresponding to 9/30/90 is subtracted from the date number for 11/14/90. The resulting value is used as the argument *Number* for the Int function.

Examples

Cut (A1:C10); removes everything from cells A1 through C10 and stores it in the pasteboard.

Cut (3); removes object number 3 from the spreadsheet and stores it in the pasteboard.

D2R (Degrees to Radians)

D2R (*Number*)

The *D2R* function converts a number from degrees to radians, according to the following formula:

Number * π / 180

D2R is particularly useful with mathematical functions, such as Acos, Asin, Atan, Atan2, Cos, Cot, Sin, and Tan.

Argument	Description
Number	Specifies the value to be converted from degrees to radians. Acceptable entries are: A number A formula that evaluates to a numeric value A cell or range reference A named range

Example

Sin (D2R (30)) gives 0.5 radians, which is equivalent to **Sin (Pi ( ) 6)**.

Date

Date (*Year, Month, Day*)

The *Date* function returns the date number of a specified date. The date number is the ParaSheet identification number for that particular date. For more information on date numbers, see the discussion of *DNumber* in the "Time Functions" section on page 54 of this manual. If you enter a date

	A cell or range
	Any combination of the above

Examples

Count (E1:E5) gives 3 when cells E1 through E3 contain numeric values, and cells E4 and E5 are blank.

Count (A1:A5, 3, 4, 5, "stop") gives 8 when cells A1 through A5 contain numeric values.

Count (F5) gives 0 when cell F5 is blank, or contains text or an error.

Cut

Cut ([*Selection*])

The *Cut* command removes an object or multiple objects, or the formatting attributes and contents (including data and formulas) from a cell, range of cells, or chart text object, and puts the information in the pasteboard. This command is equivalent to the *Cut* command in the Edit menu.

Argument	Description
Selection	Specifies the object(s) to be removed, or the cell, range of cells, or chart text object from which attributes and contents will be removed. If you don't provide a selection, ParaSheet cuts the current selection. You cannot cut objects and cell contents at the same time.

When cutting the contents of a chart text object, you must first select the chart text object using the *ChartSelect* command. Then, use the *TextSelect* command to select its contents. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object or label are already selected.)

Cut will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Cot (Cotangent)

Cot (*Angle*)

The *Cot* function returns the cotangent of an angle in radians. To convert a degree value to radians, use the D2R function or multiply it by pi/180, as shown in the example below.

Argument	Description
Angle	Specifies the size of an angle in radians. Acceptable entries are: A numeric value (formula or constant) excluding –∞ and +∞ A cell that contains a numeric value A range, the top left cell of which contains a numeric value

Examples

Cot (Pi ( ) / 4) gives 1.

Cot (Pi ( ) / 2) gives 0.

Cot (D2R (45)) gives 1 (45 degrees is equivalent to pi/4 radians).

Cot (45 * Pi ( ) / 180) gives 1 (45 degrees is equivalent to pi/4 radians).

Count (Count of Values)

Count (*List*)

The *Count* function returns the total number of numeric values in a specified list.

Argument	Description
List	Specifies the items to be counted. If you specify multiple items, they must be separated by commas. Acceptable entries are: One or more numeric or text values (formulas or constants)

at the same time.If you copy an object, ParaSheet assigns it a new object number and doesn't copy the object's name.

When copying the contents of a chart text object, you must first select the chart text object using the *ChartSelect* command. Use the *TextSelect* command to select its contents. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object or label are already selected.)

Examples

Copy (A1:C10); duplicates everything in cells A1 through C10 and stores it in the pasteboard.

Cos (Cosine)

Cos (*Angle*)

The Cos function returns the cosine of an angle given in radians. Its value is always between –1 and 1, inclusive. To convert a degree value to radians, use the D2R function or multiply it by pi/180, as shown in the example below.

Argument	Description
Angle	Specifies the size of an angle in radians. Acceptable entries are: A numeric value (formula or constant) excluding –∞ and +∞ A cell that contains a numeric value A range, the top left cell of which contains a numeric value

Examples

Cos (Pi ( )) gives –1.

Cos (D2R (60)) gives 0.5.

Cos (60 * Pi ( ) / 180) gives 0.5.

Compound (Compound Interest)

Compound (*Rate, NumPeriods*)

The *Compound* function returns the amount of interest being paid on both the interest accrued and the principal, according to the following formula:

$(1 + Rate)^{\text{NumPeriods}}$

Argument	Description
Rate	A number specifying the interest rate
NumPeriods	A number specifying the number of payment periods
	The limits for these arguments are as follows:
	If $Rate$ = –1 and NumPeriods < 0, Compound gives !NUM.
	If $Rate$ = –1 and NumPeriods = 0, Compound gives !NUM.
	If $Rate$ = –1 and NumPeriods > 0, Compound gives 0.

Examples

Compound (5%, 10) gives 1.628895.

Compound (10%, 36) gives 30.912681.

Copy

Copy ([*Selection*])

The *Copy* command duplicates the formatting attributes and contents (including data and formulas) of a range of cells, objects (including attributes), text (including attributes) in chart text objects, or chart labels, and places the duplicated information in the pasteboard. This command is equivalent to the *Copy* command in the Edit menu.

Argument	Description
Selection	Specifies the cell, range of cells, or objects to be copied. If you don't specify a selection, ParaSheet copies the current selection. You cannot copy cells and objects

cleared. If you don't provide a selection, ParaSheet clears the current selection.

ClearSpecial will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Example

ClearSpecial ("Format", SalesTotal); removes the formatting but not the data in the named range SalesTotal.

Close

Close ([*Filename*])

The *Close* command closes a spreadsheet window. This command is equivalent to the *Close* command in the Document menu. If there are any open macros for the spreadsheet, they are first automatically saved and closed.

Argument	Description
Filename	Specifies the name of the spreadsheet to be closed. If the spreadsheet is in another folder, you must provide a path name. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename (including path name) must be surrounded by quotation marks. If you don't provide a filename, ParaSheet closes the currently active spreadsheet.

Close will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Example

Close ("Energy"); closes the spreadsheet named Energy.

clear chart text objects with this command (see the ChartTextDel command).

When clearing the contents of a chart text object, you must first select the chart text object using the *ChartSelect* command. Then, use the *TextSelect* command to select its contents. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object or label are already selected.)

Clear will not execute in a user-defined function run from a cell. For more information, see the "Writing a User-Defined Function" section on page 18 of this manual.

Example

Clear (SalesTotal); clears the contents of the named range SalesTotal.

See Also

The *Cut* command

ClearSpecial

ClearSpecial ([*What* [, *Selection*]])

The *ClearSpecial* command deletes the specified contents of a range of cells, which can be data, formats, or formulas. If you don't provide any arguments, the *Clear* Special panel appears. This command is equivalent to the *Clear Special* command in the Edit menu.

Argument	Description
What	Specifies the type of information to clear. Acceptable entries are: "Format" or 1 "Content" or 2 "Both" or 3 The ParaSheet default is to clear both format and contents.
Selection	Specifies the cell or range of cells from which attributes and/or contents will be

Chr (ASCII Value to Character)

Chr (*Number*)

The *Chr* function returns the character associated with the specified ASCII value. *Chr* is the inverse of the ASC function.

Argument	Description
Number	Specifies the ASCII value representing the character to be returned. Acceptable entries are: A rational number between 0 and 255, inclusive. A cell or range containing one of the rational numbers listed above. If the number you specify is not an integer, Chr uses the integer portion (Int) of the number.

Examples

Chr (97) gives "a".

Chr (64) gives "@".

Chr (D12) gives the character "2", when cell D12 contains the value 50.

Clear

Clear ([*Selection*])

The *Clear* command deletes the contents of a cell, range of cells, an object or multiple objects, or selected text in a chart text object. This command is equivalent to the *Clear* command in the Edit menu.

Argument	Description
Selection	Specifies the cell or range of cells from which contents will be cleared, or the object or objects to be deleted. If you don't provide a selection, ParaSheet clears the current selection. You cannot clear objects and cells at the same time. You also cannot

for the second item, and so on. If you specify multiple list items, they must be separated by commas. Acceptable entries are:

A numeric value (constant or formula)

A text string (in quotes)

An error value

A cell

A range

A variable

Any combination of the above

Examples

Choose (0, "East", "West", "Central") (where East, West, and Central are named ranges) displays the first value in the named range East, because East is the first argument.

Choose (2, C1, B2, D4) displays the contents of cell D4.

Choose (2, "Blue", "Red", "Green") displays the text string "Green", because it is the third argument.

GetColNum (Choose (1, A1:B5, C1:D5, E1:F5)) displays 3, the index of the column furthest left in the range C1:D5. The Choose function here gives the range C1:D5, the second argument in its *List*. This range is then used as the argument for the GetColNum command. (See the description of GetColNum.)

Choose (3, 45, 26, "Stop", 5) displays 5.

Sum (Choose (1, A1:B5, C1:D5, E1:F5)) gives the sum of **all** the values in the range C1:D5, which is the second argument. The Choose function here gives the range C1:D5, the second argument in its List. This range is then used as the argument for the Sum function.

ChartID	Specifies the name or number of the chart. If you don't provide a *ChartID*, ParaSheet creates a new chart text object in the currently selected chart.
XCoord	Specifies the number of pixels the chart text object will be placed to the right or left of its default position. If you don't specify a location, the chart text object appears at its default top center position.
YCoord	Specifies the number of pixels the chart text object will be placed below or above its default position. If you don't specify a location, the chart text object appears at its default top center position.

Examples

ChartTextNew ("Northwestern Region", "OilConsumption", 0, 92); creates a chart text object and places it 92 pixels below its default position.

ChartTextNew ("Units Sold", "SalesTotals", 1000, –1000); creates a chart text object and aligns it with the top right corner of the chart. When you specify pixel values beyond the chart border, the new chart text object is placed at the border.

Choose

Choose (*n, List*)

The *Choose* function returns the nth item in a list of arguments. If the nth item is a cell or range, *Choose* returns the cell or range reference. Typically, this reference is used to display the value in the reference cell or range. You can also use *Choose* wherever a range reference is appropriate — for example, as an argument for the *GetColNum* command and Sum function (as shown below).

Argument	Description
n	An integer between 0 and the number of arguments in the list, minus 1; specifying the value used to select the appropriate argument from the list.
List	Specifies one or more values, from which the *n*th value is chosen. List items are numbered starting at 0 for the first item, 1

ChartTextDel

ChartTextDel ([*ChartTextID* [, *ChartID*]])

The *ChartTextDel* command removes a chart text object from a chart.

Argument	Description
ChartTextID	Specifies the identification number of the chart text object to be deleted. Chart text objects are numbered in order of creation, starting with 1. If you don't specify a *ChartTextID*, ParaSheet deletes the currently selected chart text object.
ChartID	Specifies the name or number of the chart. If you don't provide a *ChartID*, ParaSheet deletes a text object from the currently selected chart.

Example

ChartSelect ("ChartText", 3, 1, "OilConsumption");

ChartTextDel ( );

Selects and deletes chart text object number 3 in the chart named OilConsumption.

ChartTextDel (1, 3); selects and deletes chart text object number 1 in chart number 3.

ChartTextNew

ChartTextNew ([*ChartText, ChartID* [, *XCoord, YCoord*]])

The ChartTextNew command adds a chart text object to a chart. If you don't provide any arguments, ParaSheet creates an empty chart text object. This command is equivalent to the *Add Chart Text* command in the Charts menu.

Argument	Description
ChartText	Specifies the text to appear in the chart text object. Chart text must be surrounded by quotation marks. If you want to create an empty chart text object, you must still include a set of double quotation marks.

	"Yes" or 1 to add this part to the current selection
	"No" or 0 to select only this part
ChartID	Specifies the name or number of a chart. If you don't provide the *ChartID*, ParaSheet selects the specified part of the currently selected chart.

When selecting multiple chart parts, don't specify the chart identification name or number in subsequent uses of the ChartSelect command. This causes ParaSheet to replace what has already been selected with the item you are trying to add to the selection.

Examples

```
ChartSelect ( "YLabel", 1, 1, "OilConsumption" );
ChartSelect ( "YLabel", 2, 1 );
ChartSelect ( "YLabel", 3, 1 );
ChartSelect ( "YLabel", 4, 1 );
Bold( "Yes" );
```

Makes a multiple selection of the first four labels on the y-axis of a chart and makes them bold.

```
Local ( LabelNum );
Select ( "OilConsumption" );
For ( LabelNum := 1; LabelNum < 5; LabelNum := LabelNum + 1 )
    ChartSelect ( "YLabel", LabelNum, 1 );
Bold ( "Yes" );
```

Using a For loop, selects each of the first four labels on the y-axis of a chart named OilConsumption and makes them bold.

ChartSelect

ChartSelect (*PartType, PartNum, AddPart* [, *ChartID*])

The *ChartSelect* command selects part of a chart. When used multiple times with the *AddPart* argument set to "Yes", it can select several parts of a chart.

Argument	Description
PartType	Specifies the type of chart part to select. Acceptable entries are: "BarTotals" or 1 (for vertical and horizontal stacked bar charts only) "BarValues" or 2 (for vertical and horizontal bar charts only) "ChartText" or 3 "DataSeries" or 4 "XLabel" or 5 (except pie charts) "YLabel" or 6 (except pie charts) "ZLabel" or 7 (for 3D charts only) "LegendBox" or 8 "LegendEntry" or 9 "Graph" or 10 "PieLabel" or 11 (for pie charts only) "Side" or 12 (for 3D charts only)
PartNum	Specifies the particular chart part of a given type to be selected. Each part in a part type (except y-axis labels) is numbered from top to bottom or left to right starting at 1, depending on its orientation. Y-axis labels are numbered from bottom to top. Chart text objects are always numbered sequentially in order of creation.
AddPart	Selects additional chart parts when ChartSelect is used multiple times in a row, or in a loop structure. (*AddPart* is ignored when you are selecting chart text objects.) Acceptable entries are:

Argument	Description
ChartType	Type of chart to be created, corresponding to chart types in the Charts submenu. Acceptable entries are: "Bar" or 1 "HBar" or 2 (horizontal bar) "SBar" or 3 (stacked bar) "HSBar" or 4 (horizontal stacked bar) "Pie" or 5 "Area" or 6 "Line" or 7 "Scatter" or 8 "HighLow" or 9 "3DBar" or 10 "3DWire" or 11 "3DArea" or 12 "3DLine" or 13
Selection	Range of cells, named range, or chart name or number. If you select a chart, it will be reformatted to the specified *ChartType*. If you don't specify a selection, ParaSheet creates a chart from the currently selected data.

Example

ChartNew ("3DBar", A2:B5); creates a 3D bar chart from data in cells A2 through B5.

Argument	Description
ChartName	Specifies a new name for the chart. The chart name must be less than 256 characters long, and surrounded by quotation marks.
LabelsFrom	Specifies the source of chart labels. Acceptable entries are: "Rows" or 1 "Cols" or 2 The ParaSheet default is to take labels from columns.
ShowWhat	Specifies whether the legend displays. This setting is not applicable to high-low-close or wireframe charts. Acceptable entries are: "ShowLegend" or 1 "HideLegend" or 2 The ParaSheet default is to display the legend.
Freeze	Specifies whether the graph size is changeable. Acceptable entries are: "Freeze" or 1 "Unfreeze" or 2 The ParaSheet default is for the graph size to be unfrozen.

Example

ChartAttribsSet ("Expenses", "Cols", "ShowLegend", 1); gives the selected chart the name Expenses, draws chart labels from each column above the charted data set, shows the legend, and freezes the graph size.

ChartNew

ChartNew (*ChartType* [, *Selection*])

The *ChartNew* command creates a new chart of the specified chart type, from data in a specified range of cells. This command is equivalent to choosing a chart icon from the icon bar.

Examples

Ceiling (–3.9) gives –3.

Ceiling (4.85) gives 5.

Ceiling (Max (2.2, 3.5, 6.8)) gives 7. The Max function here returns the value 6.8, the maximum of the arguments in its *list*. This value is then used as the argument in the Ceiling function. (See the description of the Max function.)

Ceiling (D1:D5) gives 157 when cell D1 contains 156.3.

Cell

Cell (*Column, Row*)

The *Cell* command constructs a cell reference from a pair of column and row coordinates. Cell is particularly useful in combination with other WILMA functions and commands.

Argument	Description
Column	An index number (a positive integer greater than or equal to 1, but not exceeding 16384) representing the column to select. 1 represents column A, 2 represents column B, and so on.
Row	Number of the row to select. Row must be a positive integer between 1 and 16384.

Examples

Select (Cell (4,5)); selects cell D5.

Enter ("Investment", Cell (Arg1, Arg2)); enters the text *Investment* in the cell defined by the local variables Arg1 and Arg2.

ChartAttribsSet

ChartAttribsSet (*ChartName, LabelsFrom, ShowWhat, Freeze*)

The *ChartAttribsSet* command determines the chart name, source of chart labels, presence of the legend, and whether the graph size is frozen. This command is equivalent to settings in the Common subpanel in the Chart Inspector.

BringToFront

BringToFront ([*ObjectList*])

The *BringToFront* command places the specified objects on top of other objects in the spreadsheet. This command is equivalent to the *Bring To Front* command in the Layout submenu.

Argument	Description
ObjectList	One or more object names or numbers; or a combination of names and numbers. If you don't provide an *ObjectList*, ParaSheet brings the currently selected objects to the front.

Example

BringToFront (3, "OilConsumption"); causes object number 3 and the chart named OilConsumption to appear in front of all other objects.

Ceiling (Ceiling Value)

Ceiling (*Number*)

The *Ceiling* function returns the smallest integer greater than or equal to a specified number.

Argument	Description
Number	Specifies a numeric value. Acceptable entries are: A number (including a formula or constant) A cell containing a numeric value A range, the top left cell of which contains a numeric value

When setting the border width of a chart text object, legend, graph, or 3D chart wall, you must first select the chart part using the ChartSelect command. (This is unnecessary if the chart part is already selected.)

Examples

```
ChartSelect ( "LegendBox", 1, 0, "OilConsumption" );
BorderWidth ( 1.0 );
```

This selects the graph in a chart named OilConsumption and sets its border width to the default.

Branch

Branch (*Label*)

The *Branch* statement stems to a specified label in the current macro. WILMA continues to execute macro instructions at the new location until it reaches the end of the macro.

Argument	Description
Label	A text string specifying the location at which to continue executing. The label must occur twice in the macro: once in the Branch statement, surrounded by quotation marks, and once at the branching point, followed by a colon (:). The branching point can be before or after the Branch statement. Branch labels are case sensitive.

Example

```
Branch ( "ErrorHandler" );
\\ Additional macro instructions
ErrorHandler:
```

This tells the macro to go to the label named ErrorHandler and continue executing from that point.

See Also

For more information on the Branch statement, refer to page 28 of this manual.

Argument	Description
Shade	A number between 0.0 and 1.0, indicating the gray-scale shade of the border. 0.0 is black and 1.0 is white. –1 indicates no shade — an invisible border. The ParaSheet default is 0.0.
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID*, ParaSheet applies the border shade to the currently selected object.

When setting the border shade of a chart text object, legend, graph, or 3D chart wall, you must first select the chart part using the *ChartSelect* command. (This is unnecessary if the chart part is already selected.)

Examples

ChartSelect ("LegendBox", 1, 0, "OilConsumption");

BorderShade (–1);

This selects the legend box in a chart named OilConsumption and gives it an invisible border.

BorderWidth

BorderWidth (*Width* [, *ObjectID*])

The *BorderWidth* command sets the border thickness of the selected object. This command is equivalent to the *Border Weight* setting in the Color subpanel in the Chart Inspector.

Argument	Description
Width	A number between 0.0 and 10.0, indicating the thickness of the border. 0.0 is the thinnest and 10.0 is the thickest. The ParaSheet default is 1.0.
ObjectID	Specifies the name or number of the object. If you don't provide an *ObjectID*, ParaSheet applies the border thickness to the currently selected object.

Bold

Bold (*Setting* [, *Selection*])

The *Bold* command determines whether the contents of selected cells, chart text objects, or chart labels are bold. This command is equivalent to the *Bold/Unbold* commands in the Font submenu.

Argument	Description
Setting	Specifies whether characters are bold. Acceptable entries are: "Yes" or 1 to make characters bold "No" or 0
Selection	Specifies a cell, range of cells, or named range. If you don't provide a selection, ParaSheet makes the current selection bold.

To make the contents of a chart text object bold, you must first select the chart text object using the *ChartSelect* command. Use the *TextSelect* command to select its contents. (Using ChartSelect and/or TextSelect is unnecessary if the contents of the chart text object or label are already selected.)

Examples

Bold ("Yes"); makes the current selection bold.

ChartSelect ("ChartText", 1, 0, "OilConsumption");

TextSelect (1, 21);

Bold ("Yes");

Bolds the contents of the chart text object (assuming it contains a maximum of 21 characters) in the chart named OilConsumption.

BorderShade

BorderShade (*Shade* [, *ObjectID*])

The *BorderShade* command sets the gray-scale border shade of the selected object. This command is equivalent to settings in the *Border* color well in the Color subpanel of the Chart Inspector.

AxisTicksShow

AxisTicksShow (*ShowX, ShowY* [, *ShowZ*])

The *AxisTicksShow* command specifies whether tick marks for each axis of the selected chart display. This command is equivalent to settings in the Show area of the Marking subpanel in the Chart Inspector. It is not applicable to pie charts.

Argument	Description
ShowX	Specifies whether tick marks show on the x-axis. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is to show tick marks for the x-axis.
ShowY	Specifies whether tick marks show on the y-axis. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is to show tick marks for the y-axis.
ShowZ	Specifies whether tick marks show on the z-axis. This setting applies to 3D charts only. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is to show tick marks for the z-axis.

Examples

AxisTicksShow (1, 0); displays a selected chart with tick marks on the x-axis only.

AxisTicksShow ("No", 0, "No"); displays a selected 3D chart without tick marks on any axes.

AxisTicksCalcMan

AxisTicksCalcMan (*YAxis* [, *XAxis*])

The *AxisTicksCalcMan* command specifies custom settings for tick marks on the y-axis (and for scatter charts only, the x-axis) of the selected chart. This command is equivalent to settings in the Ticks area of the Marking subpanel in the Chart Inspector. It is not applicable to pie charts. For *AxisTicksCalcMan* to have any effect, you must first disable automatic calculation of tick marks with the *AxisTicksCalcAuto* command.

Argument	Description
YAxis	Specifies three numbers for the y-axis. The first indicates how many tick marks will be on the axis. The second indicates the starting value for the range of tick marks. The third indicates the ending value for the range of tick marks.
XAxis	For scatter charts only, specifies three numbers for the x-axis. The first indicates how many tick marks will be on the axis. The second indicates the starting value for the range of tick marks. The third indicates the ending value for the range of tick marks.

When using logarithmic scaling, you must enter positive starting and ending values for manual tick marks.

Example

```
Select ( "OilConsumption" );
AxisTicksCalcAuto ( "No" );
AxisTicksCalcMan ( 12, 5, 88 );
```

Selects a chart named OilConsumption and disables automatic calculation of tick marks. Then, sets the number of tick marks to 12, with a starting value of 5 and an ending value of 88.

Example

AxisScale ("Log"); sets the y-axis scaling for a chart to logarithmic.

AxisTicksCalcAuto

AxisTicksCalcAuto (*YAxis* [, *XAxis*])

The *AxisTicksCalcAuto* command sets or resets automatic calculation of tick marks on the y-axis (and for scatter charts only, the x-axis) of the selected chart. This command is equivalent to the Automatic Calculation check box in the Marking subpanel in the Chart Inspector. It is not applicable to pie charts.

Argument	Description
YAxis	Specifies whether tick marks on the y-axis are calculated automatically. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is automatic calculation.
XAxis	For scatter charts only, specifies whether tick marks on the x-axis are calculated automatically. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is automatic calculation.

Example

```
Select ( "OilConsumption" );
AxisTicksCalcAuto ( "No" );
```

Selects a chart named OilConsumption and sets automatic calculation of tick marks to off — a necessary step before setting tick marks manually.

ShowZ	Specifies whether z-axis labels show. This setting is only applicable to 3D charts. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is to show labels.

Examples

AxisLabelsShow (1, 0); displays a selected chart with labels on the x-axis only.

AxisLabelsShow ("No", 0, "No"); displays a selected 3D chart without any labels.

AxisScale

AxisScale (*YAxis* [, *XAxis*])

The *AxisScale* command sets the type of scaling for the y-axis (and for scatter charts only, for the x-axis) of the selected chart. This command is equivalent to choices in the Axis Scaling area of the View Options subpanel in the Chart Inspector. It is not applicable to pie charts.

Argument	Description
YAxis	Sets the type of scaling to be used for the y-axis. Acceptable entries are: "Linear" or 1 "Log" or 2 "Percent" or 3 The ParaSheet default is linear scaling.
XAxis	For scatter charts only, sets the type of scaling to be used for the x-axis. Acceptable entries are: "Linear" or 1 "Log" or 2 "Percent" or 3 The ParaSheet default is linear scaling.

	"Above" or 1 "Below" or 2 "Left" or 1 (for horizontal bar and horizontal stacked bar charts) "Right" or 2 (for horizontal bar and horizontal stacked bar charts) The ParaSheet default is Below for bar charts and Left for horizontal bar charts.

Example

Select ("OilConsumption");

AxisLabelsSet ("Below", "Right");

Selects a horizontal bar chart named OilConsumption and changes axis labels from their default positions above and to the left of chart axes.

AxisLabelsShow

AxisLabelsShow (*ShowX, ShowY* [, *ShowZ*])

The *AxisLabelsShow* command specifies whether labels for the axis of the selected chart display. This command is equivalent to choices in the Show area of the Marking subpanel in the Chart Inspector. It is not applicable to pie charts.

Argument	Description
ShowX	Specifies whether x-axis labels show. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is to show labels.
ShowY	Specifies whether y-axis labels show. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is to show labels.

ShowZ	Specifies whether the z-axis grid lines show. This setting is only applicable to 3D charts. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is to show grid lines.

Examples

AxisGridShow (1, 0); displays a selected chart with x-axis grid lines only.

AxisGridShow ("No", 0, "No"); displays a selected 3D chart without any grid lines.

AxisLabelsSet

AxisLabelsSet (*LabelY* [, *LabelX*])

The *AxisLabelsSet* command specifies the side of the grid on which labels and tick marks of an axis are drawn. This command is equivalent to choices in the Axis Labels area of the Marking subpanel in the Chart Inspector. It is not applicable to pie charts.

Argument	Description
LabelY	Specifies the side of the y-axis on which labels and tick marks are drawn. Acceptable entries are: "Left" or 1 "Right" or 2 "Above" or 1 (for horizontal bar and horizontal stacked bar charts) "Below" or 2 (for horizontal bar and horizontal stacked bar charts) The ParaSheet default is Left for bar charts and Below for horizontal bar charts.
LabelX	Specifies the side of the x-axis on which labels and tick marks are drawn. Acceptable entries are:

Examples

	A	B	C
1	12	17	21
2	15	18	14
3	8	6	9
4	10	12	13
5	21	24	8
6	19	6	10

Avg (A1:A6) gives 14.166667.

Avg (A1, B2:C6, B3) gives 11.5.

Avg (B1:B4) gives 13.25.

Avg (12, "White", 6, "Red", 3, 11, "Green") gives 8, the average of 12, 6, 3, and 11. The text entries are not numeric values and so are not included in the calculation.

AxisGridShow

AxisGridShow (*ShowX, ShowY* [, *ShowZ*])

The *AxisGridShow* command specifies whether grid lines for an axis show behind the charted data in the selected chart. This command is equivalent to the check boxes in the Show area of the Marking subpanel in the Chart Inspector. It is not applicable to pie charts.

Argument	Description
ShowX	Specifies whether the x-axis grid lines show. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is to show grid lines.
ShowY	Specifies whether the y-axis grid lines show. Acceptable entries are: "Yes" or 1 "No" or 0 The ParaSheet default is to show grid lines.

	A range, the top left cell of which contains a numeric value
YNumber	Specifies a numeric value. The same rules apply as to *XNumber*. If both *XNumber* and *YNumber* are 0, ParaSheet returns 0.

Examples

Atan2 (1, 1) gives 0.785398 (pi / 4) radians.

Atan2 (–1, –1) gives –2.356194 (–3 * pi / 4) radians.

Atan2 (–1, –1) * 180 / Pi ( ) gives –135 degrees.

R2D (Atan2 (–1, –1)) gives –135 degrees.

Avg (Average Value)

Avg (*List*)

The *Avg* function returns the average (mean) of the numeric values in a list.

Argument	Description
List	Specifies a list of one or more values. The items in the list must be separated by commas. Acceptable entries are: A numeric value or values (formulas or constants) A text value or values Null values Cells or ranges containing numeric, text, or null values Any combination of the above Avg ignores text and null values.

Argument	Description
Number	Specifies the tangent for which you want to find the arctangent. Acceptable entries are: A numeric value A formula equivalent to a numeric value A cell that contains a numeric value A range, the top left cell of which contains a numeric value If the number is +∞ or –∞, ParaSheet returns the nearest approximation to +pi/2 or –pi/2.

Examples

Atan (.5) gives 0.463648 radians.

R2D (Atan (.5)) gives 26.565051 degrees.

Atan (.5) * 180 / Pi ( ) gives 26.565051 degrees.

Atan (B1:C6) gives the arctangent of the value in cell B1.

Atan2 (Arctangent — 4 Quadrants)

Atan2 (*XNumber, YNumber*)

The *Atan2* function returns the arctangent, in radians, of two specified numbers (which are coordinates determining an angle). Its value is always between –pi and +pi, inclusive. To convert the result from radians to a degree value, use the R2D function or multiply it by 180/pi, as shown in the example below.

Argument	Description
XNumber	Specifies a numeric value. Acceptable entries are: A number A formula equivalent to a numeric value A cell containing a numeric value

Asin (Arcsine)

Asin (*Number*)

The *Asin* function returns the arcsine, in radians, of a specified number (which is the sine of an angle). Its value is always between –pi/2 and +pi/2, inclusive (in quadrant IV or I). To convert the result from radians to a degree value, use the R2D function or multiply it by 180/pi, as shown in the example below.

Argument	Description
Number	Specifies the sine for which you want to find the arcsine. Acceptable entries are: A numeric value between –1 and +1 (inclusive) A formula equivalent to a value from –1 to +1 A cell that contains a value from –1 to +1 A range, the top left cell of which contains a value from –1 to +1

Examples

Asin (.5) gives 0.523599 radians.

R2D (Asin (.5)) gives 30 degrees.

Asin (.5) * 180 / Pi ( ) gives 30 degrees.

Asin (D1) gives the arcsine of the value in cell D1.

Atan (Arctangent — 2 Quadrants)

Atan (*Number*)

The *Atan* function returns the arctangent, in radians, of a specified number (which is the tangent of an angle). Its value is always between –pi/2 and +pi/2, inclusive (in quadrant IV or I). To convert the result from radians to a degree value, use the R2D function or multiply it by 180/pi, as shown in the example below.

	If *Rate* is < –1, Annuity gives !NUM If *Rate* = –1 and NumPeriods = 0, Annuity gives –1 If *Rate* = –1 and NumPeriods > 0, Annuity gives !NUM. You can also provide a cell or range, in which case ParaSheet uses the number in the top left cell of the range.
NumPeriods	An integer specifying the number of payment periods.

Examples

Annuity (8%, 5) gives 3.99271.

Annuity (1%, 10) gives 9.471305.

ASC (Character to ASCII Value)

ASC (*Character*)

The *ASC* function returns the ASCII value representing the specified character in the NEXTSTEP character set. ASC is the inverse of the Chr function.

Argument	Description
Character	Specifies a string. *Character* must be surrounded by quotation marks. Acceptable entries are: One or more characters A cell or range that contains a string, or a formula that evaluates to a string If you specify a string of multiple characters, ParaSheet only returns the ASCII value of the first character in the string.

Examples

ASC ("a") gives 97.

ASC (Choose (3, "#", "$", "@", "%")) gives 37, the ASCII value of "%". The Choose function here returns the text value "%", the third argument in its *List*. This value is then used as an argument in the ASC function.

ASC (Num2C (E5)) gives 49 when cell E5 contains 1.

Argument	Description
How	Specifies the type of alignment for the selection. Acceptable entries are: "Left" or 1 "Center" or 2 "Right" or 3 "General" or 4 The ParaSheet default is General for cell contents, and Center for chart text objects.
Selection	Specifies a cell, range of cells, or named range. If you don't provide a selection, ParaSheet aligns the current selection.

When aligning the contents of a chart text object, you must first select the chart text object using the *ChartSelect* command. Then, use the *TextSelect* command to select its contents. (Using *ChartSelect* and/or *TextSelect* is unnecessary if the contents of the chart text object or label are already selected.)

Examples

Align ("Right", B2); aligns the contents of cell B2 to the right.

Align (2, B2); centers the contents of cell B2.

Annuity

Annuity (*Rate, NumPeriods*)

The *Annuity* function returns the amount of money payable at a regular interval. The annuity is calculated from the rate and number of payment periods, according to the following formula:

$$(1 - (1 + Rate)^{-\text{NumPeriods}}) / Rate$$

Argument	Description
Rate	A number specifying the interest rate. Limitations for *Rate* are: If *Rate* = 0, Annuity gives the value specified by *NumPeriods*

Examples

Activate ("Energy"); finds the open spreadsheet named Energy in the current folder and makes it the active window.

Activate ("/me/Data/Resources/Coal"); finds the open spreadsheet named Coal in the folder /me/Data/Resources and makes it the active window.

Alert

Alert (*Title, Message*)

The *Alert* statement causes the macro to stop running and displays an alert panel with a custom title, message, and two standard buttons: *OK* and *Cancel*. Once the user responds to the alert panel, the macro continues running.

Returns	Description
1	If the user clicks *OK*
0	If the user clicks *Cancel*

Argument	Description
Title	Specifies a text string less than 20 characters long to appear as a title for the panel. The title text must be surrounded by quotation marks.
Message	Specifies a text message to appear in the panel. The text message must be less than 255 characters long, and surrounded by quotation marks.

Example

Alert ("Warning", "Total is over budget"); causes an alert panel to appear, notifying the user that the total exceeds the amount budgeted.

Align

Align (*How* [, *Selection*])

The *Align* command sets the alignment of cell contents for a selection, or the contents of a chart text object. This command is equivalent to commands in the Alignment submenu.

	A formula equivalent to a value from –1 to +1
	A cell that contains a value from –1 to +1
	A range, the top left cell of which contains a value from –1 to +1

Examples

Acos (.5) gives 1.047198 radians.

R2D (Acos (.5)) gives 60 degrees.

Acos (.5) * 180 / Pi ( ) gives 60 degrees.

Acos (C5) gives the arccosine of the value in cell C5.

Activate

Activate (*Filename*)

The *Activate* command makes a spreadsheet the active window. This command is equivalent to selecting a spreadsheet name from the Windows menu. The spreadsheet must be open (it can be miniaturized) before it can be activated.

Argument	Description
Filename	Specifies the name of the spreadsheet to be the active window. If the spreadsheet is in another folder, you must also provide a path name. If you don't provide a path name, ParaSheet assumes *Filename* is in the current working folder. If you provide a path name other than the current working folder, ParaSheet changes the working folder to the new path. The filename, including the path name, must be surrounded by quotation marks.

Activate will not execute in a user-defined function run from a cell. For more information, refer to the "Writing a User-Defined Function" section on page 18 of this manual.

Abs (Absolute Value)

Abs (Number)

The *Abs* function gives the absolute value, or positive form, of a specified number.

Argument	Description
Number	Specifies a numeric value (constant or formula), a cell containing a numeric value, or a range, the top left cell of which contains a numeric value.

Examples

Abs (–128) gives 128.

Abs (23) gives 23.

Abs (A6) gives 391 when cell A6 contains –391.

Abs (D12:E15) gives 450 when cell D12 contains –450. (Only the absolute value of the top left cell is returned.)

Abs (Sum (C1:C2)) gives 41 when cell C1 contains –53 and cell C2 contains12. The Sum function here returns the sum of the range C1:C2. This value is then used as an argument in the Abs function, which gives the absolute value of the sum.

Acos (Arccosine)

Acos (Number)

The *Acos* function returns the arccosine, in radians, of a specified number (which is the cosine of an angle). Its value is always between 0 and +pi, inclusive. To convert the result from radians to a degree value, use the R2D function or multiply it by 180/pi.

Argument	Description
Number	Specifies the cosine for which you want to find the arccosine. Acceptable entries are: A numeric value between –1 and +1 (inclusive)

	applies view angle settings to the currently selected chart.

Example

ChartNew ("3DBar");

3DViewAngleSet (15, 45, 5);

Creates a 3D bar chart, then makes data more visible by rotating the chart to the left, displaying it more from the top than the front, and making it appear farther away.

3DViewDepthSet

3DViewDepthSet (*ZDepth* [, *XDepth* [, *ChartID*]])

The *3DViewDepthSet* command defines the viewing depth of a 3D chart. This command is equivalent to the settings in the Depth area of the View Options subpanel in the Chart Inspector.

Argument	Description
ZDepth	For 3D bar charts, ZDepth affects the depth of the bars. For 3D line or area charts, ZDepth affects the thickness of the lines or areas. Acceptable entries are numbers between 1 and 100. The ParaSheet default is 50. ZDepth is not applicable to wireframe charts.
XDepth	For 3D bar charts only, XDepth affects the width of the bars. Acceptable entries are numbers between 1 and 100. The ParaSheet default is 50.
ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet applies view depth settings to the currently selected chart.

Example

ChartNew ("3DBar");

3DViewDepthSet (45, 45);

Creates a 3D bar chart and changes bars to appear thinner and narrower.

	"No" or 0 to conceal the item Display2 is not applicable to 3D, area, high-low, pie, or stacked bar charts.
ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet applies display attributes to the currently selected chart.

Example

Select (1);

2DViewSet ("Yes", "Yes");

Selects bar chart number 1, sets bars to overlap, and shows the value of each bar.

3DViewAngleSet

3DViewAngleSet (*Rotation, Elevation, Perspective* [, *ChartID*])

The *3DViewAngleSet* command defines the viewing angles and apparent viewing distance for a 3D chart. This command is equivalent to settings in the Angles area of the View Options subpanel in the Chart Inspector.

Argument	Description
Rotation	Specifies the horizontal angle (as if you were farther to the left or right) from which to view the chart. Acceptable entries are numbers between 0 and 359 (degrees). The ParaSheet default is 30.
Elevation	Specifies the vertical angle (as if you were at eye level or above) from which to view the chart. Acceptable entries are numbers between 0 and 90 (degrees). The ParaSheet default is 30.
Perspective	Specifies the apparent distance (how close or far you seem to be) from which to view the chart. Acceptable entries are numbers between 0 and 90 (degrees). The ParaSheet default is 10.
ChartID	Specifies the name or number of a chart. If you don't provide a *ChartID*, ParaSheet

2DViewSet

2DViewSet (*Display1* [, *Display2* [, *ChartID*]])

The *2DViewSet* command specifies how data is displayed, depending on the type of chart. This command is equivalent to the check boxes in the Show area of the View Options subpanel in the Chart Inspector. It is not applicable to pie, area, or 3D charts.

Argument	Description
Display1	*Bar and horizontal bar charts:* Determines whether bars overlap. The ParaSheet default is for bars not to overlap. *Stacked bar and horizontal stacked bar charts:* Determines whether bar totals show. The ParaSheet default is to conceal bar totals. *Line and scatter charts:* Determines whether points show. The ParaSheet default is to display points. Acceptable entries are: "Yes" or 1 to display the item "No" or 0 to conceal the item *High-low charts:* Determines whether stock market symbols or hash marks show. Acceptable entries are: "Stock" or 1 "Hash" or 2 The ParaSheet default is hash marks. Display1 is not applicable to pie, area, or 3D charts.
Display2	*Bar or horizontal bar charts:* Determines whether bar values show. The ParaSheet default is to conceal bar values. *Line and scatter charts:* Determines whether lines show. The ParaSheet default is to display lines. Acceptable entries are: "Yes" or 1 to display the item