Macros

COLLABORATORS

	TITLE : Macros	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 13, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Macros

1.1 TurboCalc by Michael Friedrich

TurboCalc - copyright Michael Friedrich.

Chapter twelve is the common survey over the ARexx- and MACRO instructions. For details about both languages, please refer to chapter seven "Macro/ARexx".

Normally, an instruction consists of a leading key-word followed by parameters (separated by semicolons) enclosed in brackets (e.g. DELETE(A1:C5)).

The ARexx mode allows a specification of parameters without brackets. Then, it follows the ARexx notation where the parameters are separated by blanks only.

Thus, the three following examples are allowed forms of stating parameters in functions:

FILL(0;A1:C5)

FILL 0 A1:C5

FILL(0)

FILL 0

The different number of parameters in the examples is not of importance here (for further information, please refer to the function FILL itself).

If you use the notation without brackets, texts can be entered without quotation marks, if they do not contain blanks or quotation marks themselves.

If you use brackets, the quotations marks for strings are obligatory!

Example:

MESSAGE("This is a message";"title")

MESSAGE "This is a message" "title"

MESSAGE message title

1.2 Generals

Generals

Before introducing the instruction set, here a few hints which have importance for nearly all instructions:

Many instructions refer to the menu item of the same or a similar name. In this case, the explanation of the respective instruction is limited to basic information. If you need further details about these instructions, you can refer to the chapters, which deal with the appropriate menu item.

Most of the instructions require one or more of the following standard parameters:

Numbers

Booleans

Texts

Cell/Range

Omission of some Parameters

Omission of all Parameters

1.3 Numbers

Numbers

In most cases, a function expects an integer value as parameter; its limits can be found under the respective command. You can specify them as integer or as real value (where allowed), or replace them by formulas which return one of these values.

1.4 Booleans

Booleans

They are essential for yes/no -decisions. (For example in DBSORT for ascending/descending sorting). Here, the value 0 or FALSE corresponds to NO, any other value is interpreted as YES. (If this parameter is omitted, it corresponds to YES or this flag will be ignored. For more details about the interpretation, see the respective commands).

1.5 Texts

Texts

Here you may enter any text of any length as long as it is put in quotation marks (remember: in the ARexx mode they can be omitted). In addition to that, it is possible to use any of the offered text functions.

Hint: As with the functions, quotation marks which occur in the text must be doubled: The text a"b must be written as "a""b" and the text a""b must be written as "a""""b".

For the ARexx mode the following exceptions are valid:

Strings need not to be enclosed in quotation marks, except they contain blanks or quotation marks themselves.

Then (if the text contains blanks) the text must be put in simple as well as in double quotation marks. e.g.:

MESSAGE '"This is a text"'

or the whole instruction is enclosed by inverted commas (which makes reading a lot easier!):

'MESSAGE "This is a text"'

Further information about this topic can be found in your ARexx Manual.

1.6 Cell/Range

Cell/Range

This is the last parameter of nearly every command and specifies on which range this command should have its effect. If it is omitted, the current cell/block will be used.

Range can be replaced by any cell specification (e.g. C5 or C13:25). The functions CELL or CELLABS and the AT-function (@sheet;range) are possible as well.

Hint: With the AT-function (for detailed description, see cell functions) a macro instruction can be extended to any loaded sheet without its previous activation. COLORS (3;;@database:C5) changes the color of cell C5 of the sheet named "database" (which need not to be the currently active one).

1.7 Omission of some Parameters

Omission of some Parameters

Most instructions contain parameters whose specification is not obligatory. In this case, the respective parameters can be left out (simply enter the final semicolon or the closing bracket). If there are no more parameters to follow, or if you want to specify only a certain number of them, you can close the brackets after the last desired parameter and leave out all eventual semicola, which may follow acccording to the standard notation given in this manual.

Example: Instead of COLOR(3;4;A1:C5) you may enter COLOR(3;;A1:C5). Thus, the background color will not be changed. If you enter COLOR(3), the current cell or the current block will be used and colored.

Hint: For this reason, the last parameter of nearly every instruction is a block information. If it is omitted, the instruction will refer to the current block.- if it is given, it represents the desired range.

Within ARexx the omission of parameters can be substituted by a simple "_" or # (underscore or double cross). (If this/these parameters have to be stated at the end of an instruction, you may leave them out completely).

Example:

COLOR 3 _ C5

COLOR 3 # C5

COLOR 3

1.8 Omission of all Parameters

Omission of all Parameters

A lot of instructions allow the ommision of all parameters. The execution of such a parameter-less command causes TurboCalc to open the corresponding requester, which enables the user himself to set the data (e.g. COLORS() for the display of the color-selection window).

Attention: These instructions can only be used within a macro or ARexx program with the following limitations: TurboCalc starts an own task for the screen display and the window administration to ensure working in a true multi-tasking environment (like this, e.g. windows can be moved during a sorting process in the background). The mentioned instructions require to hand over the system control to the other task. Thus, the macro or ARexx instruction need not to wait until the user presses >OK< or >Abort<; the instruction will be finished at once with the status OK. Although the window is still visible to the user, the macro is continued with its next command. Please take care of this important fact when programming macros!

1.9 Block Instructions

Block Instructions

ADD(Data;Block])

CLEAR(Data;block])

COPY([Block])

CUT([Block])

FILL(Mode;[Block])

LANGUAGE(Mode;Block)

PASTE([Block])

PASTEDATA(Mode;[Range])

REMOVE(Data;[Block])

SERIES(Type;Increment;Columns;Range)

SORT(Ascending;Direction;Cell;Range)

TRANSPOSE([Block])

1.10 ADD(Data;Block])

ADD(Data;Block])

Inserts a block with the size of the current or specified block at the top left corner of it and moves the former contents to the right/down, or adds complete rows/columns as follows:

Data: Determines, how ADD shall be carried out:

0: One (several) entire column will be inserted.

1: One (several) entire row will be inserted.

2: A block will be inserted and the former contents (left of the block) will be shifted to the right.

3: A block will be inserted and the former contents (above the block) will be shifted downwards.

All other values produce an error message!

Block: Determines the block, which shall be inserted. If this information is missing, the current block or cell will be used.

If no parameter is specified, this instruction corresponds to <Edit-Insert Cells> - The respective selection-window will be displayed. (For further information, refer to the introductory "macro-instructions" !)

Related Functions:

CUT , CLEAR , REMOVE

1.11 CLEAR(Data;block])

CLEAR(Data;block])

Deletes the following information of the current or specified block:

Data: Determines, which information should be deleted:

0=all

1=format

2=formula (values of the last calculation stay preserved)

3=values and formula (formatting stays preserved).

All other values or the omission of this parameter produce an error message!

Block: Determines the block, which should be affected by CLEAR. If this information is missing the current cell or block will be used.

If CLEAR is called parameter-less, it corresponds to <Edit-Clear> - the respective selection-window will be displayed.

Related Functions:

CUT , CLEAR , ADD

1.12 COPY([Block])

COPY([Block])

Copies the current block or the block specified by block into the internal buffer (for PASTE)

Block: Determines the block to copy. If this information is missing, the current block is used.

If called without parameters, this command corresponds to the menu item <Edit-Copy> .

Related Functions:

CUT , PASTE

1.13 CUT([Block])

CUT([Block])

Cuts out the current block or the one specified by block.

Block: Determines the block to cut. If this information is missing, the current block will be used.

If called without parameters, this command corresponds to the menu-item <Edit-Cut>.

Related Functions:

COPY , PASTE , PASTEDATA

1.14 FILL(Mode;[Block])

FILL(Mode;[Block])

Fills the current or specified block as follows:

Mode: Determines, how the filling shall be carried out.

0: Right in every row (of the block) the last value to the left is copied to all other cells.

1: Down the values on the top of the block are copied downwards column-by-column.

2: Left starting at the right margin, all values are copied to all cells to the right.

3: Top the row at the bottom determines the filling pattern.

All other values produce an error message!

Block: Determines the block, which shall be filled. If this information is missing, the current block or cell will be used.

This command corresponds to the menu item <Edit-Fill-xxx>, depending on Mode.

Related Functions:

COPY , PASTE , SERIES

1.15 LANGUAGE(Mode;Block)

LANGUAGE(Mode;Block)

This instruction corresponds to <Edit-Translate Formula> and converts the formula and macro instructions of the block into the desired language:

Mode: 0 = German, 1 = English

Block: Determines the block, which shall be affected by LANGUAGE. If this information is missing, the current block or cell will be used.

1.16 PASTE([Block])

PASTE([Block])

Inserts the block that has been copied to the buffer by CUT or COPY at the current position/block or at the position given by block. If a block is marked (or specified), only this one will be filled (the copied contents will be either repeated or cut off).

Block: Determines the block (or a cell), where the buffer shall be inserted. If it is missing, the current cursor position or the current block will be used.

If called without parameters, this command corresponds to the meu-item <Edit-Paste>.

Hint: If you want to insert the formatting or the contents only, you have to use the command PASTEDATA.

Related Functions:

CUT , COPY , PASTEDATA

1.17 PASTEDATA(Mode;[Range])

PASTEDATA(Mode;[Range])

Inserts the block that has been copied to the buffer by CUT or COPY at the current position/block or at the position given by block. If a block is marked (or specified), only this one will be filled (the copied contents will be either repeated or cut off).

Mode: Determines, which part of the copied contents shall be inserted:

0: Pastes the formatting only.

1: Pastes values only (the formulas will be removed).

2: Pastes values and formulas - the formatting will be ignored.

Block: Determines the block (or a cell), where the buffer shall be inserted. If it is missing, the current cursor position or the current block will be used.

If called without parameters, this command corresponds to the menu-item <Edit-Paste only>.

Hint: If you want to paste the entire contents, you have to use the command PASTE.

Related Functions:

CUT , COPY , PASTE

1.18 REMOVE(Data;[Block])

REMOVE(Data;[Block])

Removes all information of the current or specified block and moves the rest to the left/top, or removes entire rows/columns as follows:

Data: Determines, how the deletion should be carried out:

0: The complete column will be deleted

1: The complete row will be deleted

2: The block will be deleted and the rest (to the right of the block) will be shifted to the left.

3: The block will be deleted and the rest (below the block) will be shifted upwards.

All other values produce an error message!

Block: Determines the block, which should be affected by REMOVE. If this information is missing, the current block or cell will be used.

If called without parameters, this command corresponds to the menu-item <Edit- Remove cells> - the appropriate selection-window will be displayed.

Related Functions:

CUT , CLEAR , ADD

1.19 SERIES(Type;Increment;Columns;Range)

SERIES(Type;Increment;Columns;Range)

Type: Specifies the type and the method for the creation of a data-series:

0: Arithmetical (standard, if this parameter is omitted)

1: Geometrical

2: Day

3: Weekday

4: Month

5: Year

Increment: Indicates the value by which the start value is increased constantly.

Columns: The series will be created columnwise, if columns is TRUE (or not nought). If set to 0 or FALSE the series will be created row by row.

Range: Determines the range, which shall be filled by SERIES. If this information is missing, the current block will be used.

If called without parameters, this command corresponds to the menu item <Data-Series> - the appropriate selection-window will be displayed.

1.20 SORT(Ascending;Direction;Cell;Range)

SORT(Ascending;Direction;Cell;Range)

This command allows to sort a certain range with the following options:

Ascending: Sorts the data-sets in ascending order, if this parameter is not 0 or omitted. The value nought or FALSE causes a sorting in descending order.

Direction: Sorts the data-range according to rows (parameter not 0 or omitted) or columns (parameter set to 0 or FALSE)

Cell: Determines, according to which row/column (depending on Direction) the data-sets shall be sorted. TurboCalc sorts according to the first row/column, if this parameter is left out.

Range: Determines the block, which shall be sorted. If this information is missing, the current block/cell will be used.

If called without parameters, this command corresponds to the menu-item <Data-Sort> - the appropriate selection-window will be displayed.

1.21 TRANSPOSE([Block])

TRANSPOSE([Block])

Transposes the current or specified block, that means that the cells will be mirrored at the diagonal from the top left to the bottom on the right. (the elements on the diagonal remain unchanged).

This corresponds to the common mathematical method of transposing a matrix.

Block: Determines the block, which shall be transposed. If this information is missing the current block will be used.

Hint: The block must have rectangular dimensions (that means height = width) - otherwise the transposing is not possible and an error message will be returned.

If called without parameters, this command corresponds to the menu-item <Edit-Transpose>.

1.22 Formatting Instructions

Formatting Instructions

ALIGNMENT([Hor];[Vert];[Block])

BOX(Left;Right;Top;Bottom;[Block])

CHANGESTYLE(Num;[Block])

COLORS([Color1];[Color2];[block])

COLUMNWIDTH(Width;[Block])

FONT([Num];(Characterset];[Block])

FRAME(Left;Right;Top;Bottom;[Block])

FREEZE(Cell)

HIDE(Row;[Block])

NUMERICFORMAT(Format;[Block])

PATTERN(Number;[Block])

PROTECTION([Write];[Formula];[Block])

ROWHEIGHT(Height;Block)

SHOW(Row;[Block])

STDFONT(Character set)

1.23 ALIGNMENT([Hor];[Vert];[Block])

ALIGNMENT([Hor];[Vert];[Block])

Determines the alignment of the current or specified block as follows:

Hor: 0=standard, 1=left, 2=right, 3=centered (all other values or omitting of the parameter do not change the horizontal alignment).

Vert: 0=standard, 1=top, 2=middle, 3=bottom (all other values or omitting of the parameter do not change the vertical alignment).

Block: Determines the block whose formatting shall be changed. If this information is missing, the current block will be used.

If called without parameters, this command corresponds to the menu-item <Format-Alignment> - the appropriate selection-window will be displayed.

Example:

ALIGNMENT(3) centers all cells of the current block

ALIGNMENT(;1;A1:A10) The contents of cell A1 to A10 will be set on "top".

ALIGNMENT() a window appears, where the user himself may select the formatting.

Related Functions:

FONT , NUMERICFORMAT , COLORS ; FRAME

1.24 BOX(Left;Right;Top;Bottom;[Block])

BOX(Left;Right;Top;Bottom;[Block])

Determines the frame around the current or specified block as follows (only the margins of the block will be changed!):

Left, right, top, bottom: Specifies the frame thickness for the respective margin:

0= off

1= thin

2= medium

3= thick

If the parameter is omitted the respective frame will not be changed.

Block: Determines the block whose formatting shall be changed. If this information is missing, the current block will be used.

This function corresponds to FRAME, but only the relevant formatting at the margin of the block will be changed.

Example:

BOX(;;3;3;A10:C20) - block A10:C20 receives a top and bottom margin (the rest remains unchanged)

BOX(1;1;1;1;A1:A10) - the cells A1 to A10 receive a thin frame on all sides.

Related Functions:

FRAME , COLORS , FONT , CHANGESTYLE , ALIGNMENT , NUMERICFORMAT

1.25 CHANGESTYLE(Num;[Block])

CHANGESTYLE(Num;[Block])

Changes the text style of the current or specified block as follows:

Num: Determines the style, which shall be applied for the current text: 1=underlined, 2=bold, 4=italic - all combinations can be reached by adding the appropriate features (e.g.: bold, italic and underlined = 2+4+1 = 7).

Hint: In contrast to FONT, which sets a charcater-set, CHANGESTYLE toggles the characteristics of any font. If the style feature was active before, it will be switched off, otherwise switched on.

Block: Determines the block whose formatting shall be changed. If this information is missing, the current block will be used.

Example:

CHANGESTYLE(3) Changes the style features bold and underlined of the current block.

CHANGESTYLE(5) switches off underlined again and switches on italic (so: bold and italic).

Related Functions:

FONT , ALIGNMENT , NUMERICFORMAT

1.26 COLORS([Color1];[Color2];[block])

COLORS([Color1];[Color2];[block])

Determines the text- and background color of the current or specified block as follows:

Color1: Determines the text color, 0=standard, 1,2,3...are further colors, refer to <Format-Colors> (-1 or omission of the parameter leaves the color unchanged).

Color2: Determines the background color, with the same options as Color1.

Block: Determines the block whose formatting shall be changed. If this parameter is missing, the current block will be used.

If called without parameters, this command corresponds to the menu-item <Format-Colors> - the appropriate selection-window will be displayed.

Example:

COLOR(3;A1:A10) - the contents of the cells A1 to A10 become blue (ref. to standard colors)

COLOR(;3) determines a new background color for the current cell.

Related Functions:

FONT , CHANGESTYLE , ALIGNMENT , NUMERICFORMAT , FRAME

1.27 COLUMNWIDTH(Width;[Block])

COLUMNWIDTH(Width;[Block])

Determines the column width of the current or specified block as follows:

Width: Determines the width of the column in characters (decimal point numbers can be entered as well)- if the value is too big or too small, it will be ignored! (Except 0, which selects the standard width!)

Block: Determines the block whose formatting shall be changed. If this information is missing, the current block will be used.

If called without parameters, this command corresponds to the menu-item <Format-Column Width> - the appropriate selection-window will be displayed.

Example:

COLUMNWIDTH(10.5) - the current columns (that means all partly marked columns) are set to a width of 10 (and a half) characters

COLUMNWIDTH(0;A1) - column A receives the standard width.

COLUMNWIDTH() - opens a window, where the user himself can set his preferred formatting.

Related Functions:

ROWHEIGHT

1.28 FONT([Num];(Characterset];[Block])

FONT([Num];(Characterset];[Block])

Determines the text style and the font of the currentor specified block as follows:

Num: Determines the style, in which the text shall appear. 0=normal, 1=underlined, 2=bold, 4=italic - furthermore, all combinations can be reached by adding the appropriate features (e.g.: bold, italic and underlined = 2+4+1 = 7) (-1 or omission of the parameter keeps the previous formatting)

Characterset: Specifies, which character set shall be used. This is a text, which consists of the character-set name (with or without the file-tag ".FONT") and a slash followed by the size (e.g. "topaz/8" or "Helvetica.font/13"). If this parameter is missing, the current character set will not be changed. The replacement by double quotation marks "" selects the standard character-set.

Block: Determines the block whose formatting shall be changed. If this information is missing, the current block will be used.

If called without parameters, this command corresponds to the menu-item <Format-Font> - the appropriate selection-window will be displayed.

Example:

FONT(3) reformats the texts of the current blocks to bold and underlined.

FONT(;"Times/13";A1:A10). the contents of cell A1 to A10 will be presented in the character set "Times", size 13 - the style remains unchanged.

FONT() opens a window, where the user hinself can set his preferred formatting.

Related Functions:

COLORS , FRAME , CHANGESTYLE , ALIGNMENT , NUMERICFORMAT

1.29 FRAME(Left;Right;Top;Bottom;[Block])

FRAME(Left;Right;Top;Bottom;[Block])

Determines the frame for the current or specified block.

Left, right, top, bottom: specifies the frame thickness for the respective margin:

0= off

1= thin

2= medium

3= thick

If the parameter is omitted the respective frame will not be changed.

Block: Determines the block, which shall be framed. If this information is missing, the current block will be used.

If called without parameters, this command corresponds to the menu-item <Format-Frame> - the appropriate selection-window will be displayed.

Example:

FRAME(;;3;3;A10) - the cell A10 receives a top and bottom frame.

FRAME(1;1;1;1;A1:A10) - the cells A1 to A10 receive a thin frame on all sides.

Related Functions:

BOX , COLORS , FONT , CHANGESTYLE , ALIGNMENT , NUMERICFORMAT

1.30 FREEZE(Cell)

FREEZE(Cell)

Freezes all rows above Cell and all columns left to Cell.

Cell: Is the start cell of the freeze operation according to the rules given above. If this parameter is missing, the current cell is used.

If called without parameters, this instruction corresponds to <Format-Freeze>. Further details can be found there.

Example:

FREEZE(C2) - column A and B and row 1 are frozen and will not be scrolled with the rest of the sheet but will stay fixed on the screen.

FREEZE(A1) - unfreezes all previous frozen columns/rows.

1.31 HIDE(Row;[Block])

HIDE(Row;[Block])

Hides the current column or row:

Row: Determines, if TurboCalc should discontinue displaying the specified row or column:

0: Column

1: Row

Block: Determines the block, the instruction shall be used for (depeding on the value of Row only the row or column will be used). If this information is missing, only the current block or -cell will be used.

If called without block-parameter, this command corresponds to the menu-items <Format-Hide-Column> or <Format-Hide-Row> - the appropriate selection-windows will be displayed.

Example:

HIDE(0) hides the current column.

HIDE(1;C2:C3) hides row 2 and 3.

1.32 NUMERICFORMAT(Format;[Block])

NUMERICFORMAT(Format;[Block])

Determines the numeric format of the current or specified block as follows:

Format: 0=standard, number 1 to 40 represent the respective numeric format (beginning with 0= standard) as follows:

0 standard

1 0

2 0,0

...

7 0,000000

8 0,000

9 0.000,00

10 000

11 0000

12 00000

13 0,0E+00

...

18 0,000000E+00

19 0%

...

25 0,000000%

26 $0

27 $0,00

28 $0,000

29 $0.000,00

30 DD.MM.YYYY

31 DD.MM.YY

32 DD.MMM.YYYY

33 DD.MMM YY

34 DD.MMM

35 MMM.YYYYY

36 MMM.YY

37 hh:mm:ss

38 hh:mm

39 h:mm:ss

40 h:mm

Block: Determines the block whose formatting shall be changed. If this information is missing, the current block or -cell will be used.

If called without parameters, this command corresponds to the menu-item <Format - Numeric Format> - the appropriate selection-window will be displayed.

Example:

NUMERICFORMAT(0) the current cells are presented in standard formatting.

NUMERICFORMAT(2;A1:A10) sets the format "0.00" for cells A1 to A10

NUMERICFORMAT() opens a window, where the user hinself can set his preferred formatting.

Related Functions:

FONT , ALIGNMENT , COLORS , FRAME , BOX

1.33 PATTERN(Number;[Block])

PATTERN(Number;[Block])

Determines the background pattern of the current or specified block.

The parameter Number ranges between 0 and 15 and determines one of 15 possible background patterns. The optional parameter Block determines the range whose background pattern shall be changed.

Without paramaters this command corresponds to <Format-Pattern>.

Related Functions:

COLORS

1.34 PROTECTION([Write];[Formula];[Block])

PROTECTION([Write];[Formula];[Block])

Determines the protection of the current or specified block as follows:

Write: Determines, if TurboCalc grants the write access for these cells. (0=yes, 1=no, -1=do not change.)

Formula: Determines, if possible formulas within these cells shall be displayed or hidden (0=display, 1=hide, -1=do not change)

Block: Determines the block whose access privileges shall be changed. If this information is missing, the current block will be used.

If called without parameters, this command corresponds to the menu-item <Format-Protection> - the appropriate selection-window will be displayed.

Example:

PROTECTION(0) despite of a previous protection of the sheet, this command allows editing the current cells.

PROTECTION(;1;A1:A10) locks the display of formulas within the block A1:A10.

PROTECTION() opens a window, where the user hinself can set his preferred formatting.

Related Functions:

FONT , ALIGNMENT , NUMERICFORMAT , COLORS , FRAME , BOX

1.35 ROWHEIGHT(Height;Block)

ROWHEIGHT(Height;Block)

Determines the row height of the current or specified block as follows:

Height: Determines the height of the row according to the height of the current font, decimal values are allowed as well - if the value is too big or too small, it will be ignored! (Except 0, which selects the standard width!)

Block: Determines the block whose formatting shall be changed. If this information is missing, the current block will be used.

If called without parameters, this command corresponds to the menu-item <Format-Row Height> - the appropriate selection-window will be displayed.

Example:

ROWHEIGHT(4) the current rows (that means all partly marked rows) are set to a height of 4 times the font height.

ROWHEIGHT(1.5;A1) row 1 receives a height of 1.5 rows.

ROWHEIGHT() opens a window, where the user hinself can set his preferred formatting.

Related Functions:

COLUMNWIDHT

1.36 SHOW(Row;[Block])

SHOW(Row;[Block])

Redisplays hidden rows or columns:

Row: Determines, if TurboCalc should discontinue hiding the specified row or column:

0: Column

1: Row

Block: Determines the block, the instruction shall be used for (depeding on the value of Row only the row or column will be used). If this information is missing, only the current block or -cell will be used.

If called without second parameter, this command corresponds to the menu-items <Format-Show-Column> or <Format-Show-Row> - the appropriate selection-windows will be displayed.

Example:

SHOW(0) rediplays the current column.

SHOW(1;C2:C3) redisplays the rows 2 and 3.

Related Functions:

HIDE

1.37 STDFONT(Character set)

STDFONT(Character set)

This function determines the standard character-set for the respective sheet.

Character set: is a character-set text (form: name/size, e.g. "Helvetica/13"). If this parameter is not specified, a selection window will appear.

If called without parameters, this command corresponds to the menu-item <Format - Standard Font> - the appropriate selection-window will be displayed.

1.38 Cursor Control

Cursor Control

COLUMN(Column)

CURRENTCELL()

GOTOCOLUMN(Column)

GOTOLINE(Line)

LASTCOLUMN()

LASTROW()

LEFT(Num

RIGHT(Num

UP(Num

DOWN(Num)

LINE(Line)

FIND(Text;Part;Case;Columns;Range)

SELECT([Block])

1.39 COLUMN(Column)

COLUMN(Column)

Moves the cursor within current sheet column columns to the left or right (1 means one column to the right).

Related Functions:

GOTOCOLUMN , GOTOLINE , LINE , UP, DOWN , LEFT, RIGHT, SELECT

1.40 CURRENTCELL()

CURRENTCELL()

Shifts the visible part of the screen, so that the current cell or - block gets visible (if it is already visible, this instruction has no effect).

If called without parameters, this command corresponds to the menu-item <Command-Show Current Cell>.

Related Functions:

SELECT

1.41 GOTOCOLUMN(Column)

GOTOCOLUMN(Column)

Moves the cursor within the current sheet to the column Column (1 represents the column totally on the left of the sheet).

Related Functions:

GOTOLINE , COLUMN , LINE , UP, DOWN , LEFT, RIGHT, SELECT

1.42 GOTOLINE(Line)

GOTOLINE(Line)

Moves the cursor within the current sheet to the line Line (1 represents the line on the top of the sheet).

Related Functions:

GOTOCOLUMN , COLUMN , LINE , UP, DOWN , LEFT, RIGHT, SELECT

1.43 LASTCOLUMN()

LASTCOLUMN()

Moves the cursor, starting at its current position, to the last column, which is filled with data (in this line).

Related Functions:

GOTOCOLUMN , GOTOLINE , LINE , LASTLINE, UP, DOWN , LEFT, RIGHT, SELECT

1.44 LASTROW()

LASTROW()

Moves the cursor, starting at its current position, to the last row, which is filled with data (in the current column)

Related Functions:

COLUMN , LASTCOLUMN , LINE , GOTOLINE , UP, DOWN , LEFT, RIGHT, SELECT

1.45 LEFT(Num

LEFT(Num

Moves the cursor by Num steps to the appropriate direction. If Num is missing, the cursor will be moved by one field.

Num: Can be any (positive) number. If it is missing, 1 will be assumed.

Attention: LEFT or RIGHT cannot be used as a macro instruction (only as ARexx command), because functions with the same name (but with other effects and other parameters) are already existing. In macros you have to use COLUMN(-1) or COLUMN(1)

Related Functions:

LINE , COLUMN , GOTOLINE , GOTOCOLUMN , SELECT , RIGHT, UP, DOWN

1.46 RIGHT(Num

RIGHT(Num

Moves the cursor by Num steps to the appropriate direction. If Num is missing, the cursor will be moved by one field.

Num: Can be any (positive) number. If it is missing, 1 will be assumed.

Attention: LEFT or RIGHT cannot be used as a macro instruction (only as ARexx command), because functions with the same name (but with other effects and other parameters) are already existing. In macros you have to use COLUMN(-1) or COLUMN(1)

Related Functions:

LINE , COLUMN , GOTOLINE , GOTOCOLUMN , SELECT , LEFT, UP, DOWN

1.47 UP(Num

UP(Num

Moves the cursor by Num steps to the appropriate direction. If Num is missing, the cursor will be moved by one field.

Num: Can be any (positive) number. If it is missing, 1 will be assumed.

Related Functions:

LINE , COLUMN , GOTOLINE , GOTOCOLUMN , SELECT , LEFT, RIGHT, DOWN

1.48 DOWN(Num)

DOWN(Num)

Moves the cursor by Num steps to the appropriate direction. If Num is missing, the cursor will be moved by one field.

Num: Can be any (positive) number. If it is missing, 1 will be assumed.

Related Functions:

LINE , COLUMN , GOTOLINE , GOTOCOLUMN , SELECT , LEFT, RIGHT, UP

1.49 LINE(Line)

LINE(Line)

Moves the cursor within the current sheet line rows up or down (1 means one row downwards).

Related Functions:

COLUMN , GOTOLINE , GOTOCOLUMN , UP, DOWN , LEFT, RIGHT, SELECT

1.50 FIND(Text;Part;Case;Columns;Range)

FIND(Text;Part;Case;Columns;Range)

Searches for the next appearance of text and locates the cursor in this cell.

Text: The text which shall be found.

Part: Determines, whether the cell must contain the exact text (0 or FALSE), or whether a part-congruity is sufficient (TRUE or not nought). If this parameter is omitted, TRUE will be assumed.

Case: Determines the case sensitivity. Nought or FALSE means, that even capitalization must be congruent. TRUE or not nought means, that a simple accordance of the single letters is sufficient. If this parameter is omitted, TRUE will be assumed.

Columns: Determines, if the text shall be searched columnwise (0 or FALSE) or linewise (TRUE or not nought) . If this parameter is omitted, a linewise proceeding will be assumed.

Range: If a cell is specified here, it determines the starting position of search.

If a range is specified here, the search will be started in the top left corner of it and the procedure is limited to this range.

If called without parameters, this command corresponds to the menu-item <Command-Find>.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: The cell in which this instruction is located receives the value TRUE, if the search operation has been successful. If the search has failed, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see similar example at DBFIND).

Example:

FIND("hello")

searches for the string "hello", the capitalization is not important and the text may also occur as a part of a string . (e.g. "Hello Alex").

FIND ("hello";;0)

Dito, but now the word must be written in small letters ("hello Alex").

1.51 SELECT([Block])

SELECT([Block])

(within ARexx: SELECTCELL or SELECTRANGE)

Locates the cursor to the position specified by block (if it is a single cell) or marks the block and locates the cell cursor in the top left corner of the block. If this range is not visible at the moment you call this command, the screen will be scrolled automatically.

Block: Determines the new cursor position. It an be a cell reference or a block (or, of course, a name-substitution, e.g. DATABASE)

Hint: For ARexx the synonyms SELECTCELL or SELECTRANGE have been introduced, as SELECT is a reserved key-word of ARexx, which requires to be set in quotation marks. It is advisable to use SELECTCELL there.

If called without parameters, this command corresponds to the menu-item <Command-Goto> - the appropriate selection-window will be displayed.

Related Functions:

CURRENTCELL

1.52 Input Instructions

Input Instructions

BEEP()

DELAY(Time)

INPUT(Text[;Title];[Cell])

INSERTFORMULA()

INSERTMACRO()

INSERTNAME()

MESSAGE(Text[;Title])

PUT(Contents[;Cell])

REQUEST(Text[;Title])

1.53 BEEP()

BEEP()

Produces a short flash on the screen together with a warning-signal. The audio-signal is limited to OS2.0 and higher (both can be switched on/off via Sound of the Workbench-Preferences).

BEEP is useful to indicate an error, if the user did something wrong.

1.54 DELAY(Time)

DELAY(Time)

Stops the macro-execution for a certain time, given with the parameter Time.

This command is useful in connection with the start of external Amiga-DOS commands via the RUN-instruction.

Time: specifies the delay-duration in 1/50 seconds. The maximum value ranges at 30*50=1500, which corresponds to 30 seconds. This restriction was introduced for security reasons in order to avoid an inappropriate blockade of TurboCalc.

If you need more time, you can execute several DELAY commands in a sequence.

If you omit the paramter time, the standard delay of one second (i.e. the value 50) is assumed.

Attention: The wait-state of TurboCalc can be interrupted in the macro-mode only (with the menu-item <Macro - Stop Macro>, which is checked every second second).

If DELAY was started via ARexx or directly (<F10>), you cannot interrupt. You have to wait until the specified time has passed (which is the reason for the delay restriction of this command).

1.55 INPUT(Text[;Title];[Cell])

INPUT(Text[;Title];[Cell])

Opens a window and displays the parameter text in this window. Title determines the window title. The window receives the gadgets >Ok< and >Abort< and a text gadget, which is provided for user input. A click on >OK< or pressing <Return> transfers this input to the current or specified cell.

Text: is a normal text (in quotation marks or given as a text formula). It will be printed within the window - if it is longer than one line, it will be separated automatically (the tilde "~" can be used as a help-dash: It will not be printed, except at the end of a

line, where it changes into a dash "-". The "turned P" (ASCII-Code 182, produced by <ALT>+<P>) means "end of line". After this code, the text is continued in the next line).

Title: Determines the window title. A standard title appears, if you omit this parameter.

Cell: Determines the cell the result after >OK< should be transferred to. If you omit this parameter, the current cell in the current window will be used (which is the window the macro is called from and not the macro window with its source code).

Hint: If you want to include cell contents in the text, use the function TEXT (cell) and the concatenation by "+", e.g. "contents of cell A1 is"+TEXT(A1)+"!". For further information, also refer to the third example of MESSAGE.

Hint: If you use this command in the ARexx mode and if the specified text or cell contains blanks, make sure to put them in quotations marks. You can either enclose the whole command in inverted commas or the text in inverted commas and quotation marks: e.g. ’ "This is a text" ’.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if you touch the >OK< gadget . If you click on >Abort< or close the window with the closing gadget in the top left corner, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example below).

In the ARexx Mode: For >Ok< no error message is produced, otherwise the error ABORT will be given back.

Example:

INPUT("Please enter your name";"Your Name";A1)

Opens a window with the title "Your Name" and displays "Please enter your name". The text gadget can now be filled with the user’s name. >Ok< puts it to cell A1, >Abort< ignores the input.

The following program extract realizes a name request, e.g. for security reasons:

If the user does not enter a correct text (that means: no alphanumeric data) or passes by with >Abort<, the macro main-routine is skipped. Otherwise, the message "Hello name" is displayed.

(Please note the notation "#A10" for checking the cell A10 in the macro sheet (and not in the current cell)

A10 =INPUT("Please enter your name";"Personal data";A1)

A11 =IFGOTO(NOT(#A10 AND ISSTRING(A1));A20)

A12 =MESSAGE("Hello"+A1)

... main routine ...

A20 =RETURN()

/*ARexx-Example:*/

OPTIONS RESULTS

ADDRESS TCALC

INPUT ’"Please enter your name"’ ’"Your Name"’ C5

GETVALUE C5

SAY "Hello" Result

This example asks for the name, writes it to C5 and then prints out "hello" + name.

(Please make sure not to forget the single and double quotation marks, which enclose the texts "Please enter your name" and "your name", because they contain blanks).

Instead of the INPUT line above, the following expression would also be possible (everything enclosed in single quotation marks):

’INPUT "Please enter name" "Your name" C5’

1.56 INSERTFORMULA()

INSERTFORMULA()

This corresponds to the menu item <Command-Paste-Formula> . It opens a window, where the user can select one admitted formula, which shall be inserted in the current cell.

1.57 INSERTMACRO()

INSERTMACRO()

This corresponds to the menu item <Command-Paste-Macro>. It opens a window, where the user can select one admitted macro-instruction, which shall be inserted in the current cell.

1.58 INSERTNAME()

INSERTNAME()

This corresponds to the menu item <Command-Paste-Name>. It opens a window, where the user can select a defined name, which shall be inserted in the current cell.

1.59 MESSAGE(Text[;Title])

MESSAGE(Text[;Title])

Opens a window and displays the parameter text in this window. Title determines the window title. The window receives the gadgets >More<. A click on this gadget or closing the wondow resumes the macro with its nect instruction.

Text: is a normal text (in quotation marks or given as a text formula). It will be printed within the window - if it is longer than one line, it will be separated automatically (the tilde "~" can be used as a help-dash: It will not be printed, except at the end of a line, where it changes into a dash "-". The "turned P" (ASCII-Code 182, produced by <ALT>+<P>) means "end of line". After this code, the text is continued in the next line).

Title: Determines the window title. A standard title appears, if you omit this parameter.

Hint: If you want to include cell contents in the text, use the function TEXT (cell) and the concatenation by "+", e.g. "contents of cell A1 is"+TEXT(A1)+"!". For further information, also refer to the third example).

Example:

MESSAGE("This is a test";"title")

Opens a window with the title "title" and displays "This is a test".

MESSAGE("Alongwordwith~separator").

The tilde determines at which point the text is separated.

MESSAGE("cell A1: "+TEXT(A1)+CHAR(182)+" Cell A2:"+TEXT(A2)+CHAR(182)+" A1+A2:"+TEXT(A1+A2)

CHAR(182) is the "turned P" (can also be entered directly by <Alt>-<P>), which forces TurboCalc to continue the text in a new line.

1.60 PUT(Contents[;Cell])

PUT(Contents[;Cell])

Writes the parameter contents to the specified cell (resp. the current cell). This corresponds to the direct input of this contents.

Contents: the input of the specified cell. It is treated like the direct input by using the keyboard - therefore, date-values, time-values, booleans and formulas can also be entered. The contents must be enclosed in quotation marks (If the text or the formula itself contains single quotation marks, you have to substitute them by two of the same kind (see below)).

Numbers and booleans can be enterd directly as parameter (without quotation marks), which avoids a unnecessary conversion into text. The same is valid for date- and time-values. Please note that these cannot be entered directly in formulas (only with VALUE, DATEVALUE...).

Cell: Determines the cell, which shall be filled with contents. If it is missing, the current cursor position will be used.

Example:

WRITE("12";A1)

WRITE(12;A1)

Both instructions do the same. They write the numerical value 12 to cell A1

PUT(A1+1;A1)

Increases the contents of cell A1 by 1 (very useful in connection with IFGOTO, see there)

PUT("=A1+1";A2)

Writes the formula =A1+1 to cell A2

PUT("He said:""Hello""";A1)

Writes the text "He said:"Hello"" to cell A1 (With two quotation marks!!!).

PUT("""12";A1)

Writes the alphanumerical value 12 to cell A1 (that means the text "12, so that it is not interpreted as a number).

1.61 REQUEST(Text[;Title])

REQUEST(Text[;Title])

Opens a window and displays the parameter text in this window. Title determines the window title. The window receives the gadgets >Ok< and >Abort<.

Text: is a normal text (in quotation marks or given as a text formula). It will be printed within the window - if it is longer than one line, it will be separated automatically (the tilde "~" can be used as a help-dash: It will not be printed, except at the end of a line, where it changes into a dash "-". The "turned P" (ASCII-Code 182, produced by <ALT>+<P>) means "end of line". After this code, the text is continued in the next line).

Title: Determines the window title. A standard title appears, if you omit this parameter.

Hint: If you want to include cell contents in the text, use the function TEXT (cell) and the concatenation by "+", e.g. "contents of cell A1 is"+TEXT(A1)+"!". For further information, also refer to the third example of MESSAGE.

Hint: If you use this command in the ARexx mode and if the specified text or cell contains blanks, make sure to put them in quotations marks. You can either enclose the whole command in inverted commas or the text in inverted commas and quotation marks: e.g. ’ "This is a text" ’.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if you touch the >OK< gadget . If you click on >Abort< or close the window with the closing gadget in the top left corner, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example below).

In the ARexx Mode: For >Ok< no error message is produced, otherwise the error ABORT will be given back.

Example:

REQUEST("This is a text";"Test-Window")

Opens a window with the title "Test-Window" and displays "this is a test".

You can click on >OK> or pass by with >Abort>.

The following program extract realizes a security request:

(Please note the notation "#A10" for checking the cell A10 in the macro sheet (and not in the current cell)

A10 =REQUEST("Are you sure ?";"Test-Macro";)

A11 =IFGOTO(NOT(#A10);A20)

... main routine ...

A20 =RETURN()

1.62 Load / Save

Load / Save

CSVINSERT([Block];[Name];[Separator])

CSVLOAD([Name]);[Separator])

CSVSAVE([Name]);[Separator])

CSVSAVEBLOCK([Block];[Name];[Separator])

LOAD([Name])

LOADCONFIG()

PROCALCINSERT([Block];[Name])

PROCALCLOAD([Name])

SAVE([Name])

SAVEAS([Name])

SAVEBLOCK([Block];[Name])

SAVECONFIG()

SYLKINSERT([Block];[Name])

SYLKLOAD([Name])

SYLKSAVE([Name])

SYLKSAVEBLOCK([Block];[Name])

TCDINSERT([Block];[Name])

1.63 CSVINSERT([Block];[Name];[Separator])

CSVINSERT([Block];[Name];[Separator])

Inserts the sheet name at the position block. The sheet has to be in the CSV format - for further information refer to the appendix "Sheet Formats".

Block: Determines the top left corner, where the data insertion shall start. If it is omitted, the insertion begins at the current cursor position.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Edit -Insert File - CSV >.

Related Functions:

TCDINSERT , SYLINSERT, PROCALINSERT

1.64 CSVLOAD([Name])

CSVLOAD([Name])

μ;[Separator])

Opens a new sheet (except the current sheet is new and still empty) and loads the sheet name. The sheet has to be in the CSV format - or further information refer to the appendix "Sheet Formats".

Name: Specifies the name of the file to load. Please specify the complete path (e.g. "DHO:TurboCalc/sheets/test.CSV"), if possible.

Separator: Specifies the character with which the CSV-format separates the different cell contents. If this information is missing, the semicolon ";" will be taken as default value. The character has to be put in quotation marks, e.g. "," or ";" or " " (blank). "T" represents the tab(ulator), "S" is the blank (space).

Hint: If only one parameter is specified, this will normally be the name.

Exception: If the name consists of one character only, it will be interpreted as a separator and the requester for file-selection appears.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Project - Import -CSV>.

Related Functions:

LOAD , SYLKLOAD , PROCALCLOAD

1.65 CSVSAVE([Name])

CSVSAVE([Name])

μ;[Separator])

Saves the current sheet in the CSV format under the name name - for further information about the file format refer to the appendix "Sheet Formats".

Name: Specifies the name of the file under which the sheet shall be saved. Please specify the complete path (e.g. "DHO:TurboCalc/sheets
if possible.

Separator: Specifies the character with which the CSV-format separates the different cell contents. If this information is missing, the semicolon ";" will be taken as default value. The character has to be put in quotation marks, e.g. "," or ";" or " " (blank). "T" represents the tab(ulator), "S" is the blank (space).

Hint: If only one parameter is specified, this will normally be the name.

Exception: If the name consists of one character only, it will be interpreted as a separator and the requester for file-selection appears.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Project - Export -CSV>.

Related Functions:

SAVE , SAVEAS , SYLKSAVE

1.66 CSVSAVEBLOCK([Block];[Name];[Separator])

CSVSAVEBLOCK([Block];[Name];[Separator])

Saves the current or specified block of the current sheet as name in the CSV-format (see appendix "Sheet Formats"). If this name is missing, a file requester for name-selection will be displayed.

Block: Determines the block to save.

Name: Specifies the name of the file under which the sheet shall be saved. Please specify the complete path (e.g. "DHO:TurboCalc/sheets
if possible.

Separator: Specifies the character with which the CSV-format separates the different cell contents. If this information is missing, the semicolon ";" will be taken as default value. The character has to be put in quotation marks, e.g. "," or ";" or " " (blank). "T" represents the tab(ulator), "S" is the blank (space).

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Edit -Save Block as - CSV >.

Related Functions:

SAVE , SAVEAS , CSVSAVE , SYLKSAVE

1.67 LOAD([Name])

LOAD([Name])

Opens a new sheet (except the current sheet is new and still empty) and loads the sheet name. The sheet has to be in the TurboCalc standard format - or further information, refer to the appendix "Sheet Formats".

Name: Specifies the name of the file to load. Please specify the complete path (e.g. "DHO:TurboCalc/sheets/test.TCD"), if possible.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Project-Open> .

Related Functions:

CSVSAVE , SYLKLOAD , PROCALCLOAD

1.68 LOADCONFIG()

LOADCONFIG()

This corresponds to the menu item <Options-Config-Load> and searches for the file "S:TurboCalc.CFG". If the file was found, the current settings are changed accordingly.

1.69 PROCALCINSERT([Block];[Name])

PROCALCINSERT([Block];[Name])

Inserts the sheet name at the position block. The sheet has to be in the ProfessionalCalc file-format - for further information refer to the appendix "Sheet Formats".

Block: Determines the top left corner, where the data insertion shall start. If it is omitted, the insertion begins at the current cursor position.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Edit-Insert File-ProCalc>

For details see instruction PROCALCLOAD (which is identical, except of the first parameter).

Related Functions:

TCDINSERT , CSVINSERT , SYLKINSERT

1.70 PROCALCLOAD([Name])

PROCALCLOAD([Name])

Opens a new sheet (except the current sheet is new and still empty) and loads the sheet name. The sheet has to be in the ProfessionalCalc file-format - for further information refer to the appendix "Sheet Formats".

Name: Specifies the name of the file to load. Please specify the complete path (e.g. "DHO:TurboCalc/sheets/test.PCF"), if possible.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item<Project-Import-ProCalc>.

Related Functions:

LOAD , CSVLOAD , SYLKLOAD

1.71 SAVE([Name])

SAVE([Name])

Saves the current sheet in the TurboCalc standard format - for further information refer to the appendix "Sheet Formats".

Name: Specifies the name of the file to save. Please specify the complete path (e.g. "DHO:TurboCalc/sheets/test.TCD"), if possible.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Project-Save> .

Related Functions:

SAVEAS , CSVSAVE , SYLKSAVE

1.72 SAVEAS([Name])

SAVEAS([Name])

Saves the sheet in the standard TurboCalc-format. In contrast to the SAVE instruction, this command always opens a file-requester for name-selection. If the parameter name is given, this will be taken as default save-name, otherwise the name of the current sheet will be used.

Name: Specifies the default name for the file requester. Please specify the complete path (e.g. "DHO:TurboCalc/sheets/test.TCD"), if possible.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Project-Saveas>.

Related Functions:

SAVE , CSVSAVE , SYLKSAVE

1.73 SAVEBLOCK([Block];[Name])

SAVEBLOCK([Block];[Name])

Saves the current or specified block of the current sheet as name in the standard TurboCalc-format (see appendix, "Sheet Formats"). If this name is missing, a file requester for name-selection will be displayed.

Block: Determines the block to save.

Name: Specifies the name of the file under which the sheet shall be saved. Please specify the complete path (e.g. "DHO:TurboCalc/sheets
if possible.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Edit - Save Block as -TurboCalc>.

Related Functions:

SAVE , SAVEAS , CSVSAVE , SYLKSAVE

1.74 SAVECONFIG()

SAVECONFIG()

This corresponds to the menu item <Options-Config-Save> and saves the current settings as "S:TurboCalc.CFG" - this file is loaded after every start of TurboCalc or <Project-New> and the corresponding settings are taken over.

The current cell contents are not saved with this instruction.

Hint: The file "S:TurboCalc.CFG" is a normal TurboCalc data-file as it is produced by saving with <Project-Save as>. Thus, if you want to define a "Default-Sheet" that shall be used each time <Project-New> is called (or during startup), simply save your desired default sheet as "S:TuboCalc.CFG" with <Project-Save as>.

1.75 SYLKINSERT([Block];[Name])

SYLKINSERT([Block];[Name])

Inserts the sheet name at the position block. The sheet has to be in the SYLK-format - for further information refer to the appendix "Sheet Formats".

Block: Determines the top left corner, where the data insertion shall start. If it is omitted, the insertion begins at the current cursor position.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Edit-Insert File-SYLK>.

For details see instruction SYLKLOAD (which is identical, except of the first parameter).

Related Functions:

TCDINSERT , CSVINSERT , PROCALINSERT

1.76 SYLKLOAD([Name])

SYLKLOAD([Name])

Opens a new sheet (except the current sheet is new and still empty) and loads the sheet name. The sheet has to be in the SYLK-format - for further information refer to the appendix "Sheet Formats".

Name: Specifies the name of the file to load. Please specify the complete path (e.g. "DHO:TurboCalc/sheets/test.SLK"), if possible.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Project - Import from -SYLK>.

Related Functions:

LOAD , SYLKINSERT

1.77 SYLKSAVE([Name])

SYLKSAVE([Name])

Saves the current sheet as name in the SYLK-format (see appendix, "Sheet Formats"). If this name is missing, a file requester for name-selection will be displayed.

Name: Specifies the name of the file under which the sheet shall be saved. Please specify the complete path (e.g. "DHO:TurboCalc/sheets
if possible.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Project - Export to -SYLK>.

Related Functions:

SAVE , SAVEAS , CSVSAVE ,

1.78 SYLKSAVEBLOCK([Block];[Name])

SYLKSAVEBLOCK([Block];[Name])

Saves the current or specified block of the current sheet as name in the SYLK-format (see appendix, "Sheet Formats"). If this name is missing, a file requester for name-selection will be displayed.

Block: Determines the block to save.

Name: Specifies the name of the file under which the sheet shall be saved. Please specify the complete path (e.g. "DHO:TurboCalc/sheets
if possible.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Project - Save Block as -SYLK>.

Related Functions:

SAVE , SAVEAS , CSVSAVE , SYLKSAVE

1.79 TCDINSERT([Block];[Name])

TCDINSERT([Block];[Name])

Inserts the sheet name at the position block. The sheet has to be in the stadard TurboCalc-format (TCD is the abbrevation of TurboCalcData)- for further information refer to the appendix "Sheet Formats".

Block: Determines the top left corner, where the data insertion shall start. If it is omitted, the insertion begins at the current cursor position.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if loading was finished succesfully . If an error occured during the process or the user aborted the loading/saving, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example at DBFIND).

If called without parameters, this command corresponds to the menu-item <Edit-Insert File-TurboCalc>.

For details see instruction LOAD (which is identical, except of the first parameter).

Related Functions:

CSVINSERT , SYLKINSERT , PROCALCINSERT

1.80 Database Instructions

Database Instructions

CRITERIA([Range])

DATABASE([Range])

DBDELETE()

DBEXTRACT([Cell])

DBFIND([Cell])

DBSORT(Ascending;[Cell])

1.81 CRITERIA([Range])

CRITERIA([Range])

Determines the new range, which serves as sort criterion (corresponds to <Data-Define criteria>).

If the parameter range is omitted, the currently marked block will be used.

Further information about the creation of search criteria can be found in chapter four "Database".

Example:

CRITERIA() - determines the current block as new search criteria.

CRITERIA(A1:C10) - the block A1 to C10 is the new search-criteria-range.

Related Functions:

DATABASE , DBFIND , DBEXTRACT , DBDELETE , DBSORT

1.82 DATABASE([Range])

DATABASE([Range])

Determines the new database range (corresponds to <Data-Define Database>).

If the parameter range range is omitted, the currently marked block will be used.

For details about database ranges, refer to chapter four "Database".

Example:

DATABASE() - determines the current block as a new database range

DATABASE(A1:C10) - block A1:C10 is the new database range

Related Functions:

CRITERIA , DBFIND , DBEXTRACT , DBDELETE , DBSORT

1.83 DBDELETE()

DBDELETE()

Deletes all data sets which correspond to the current criteria (corresponds to <Data-Delete>).

For details, refer to chapter four "database".

Example:

DBDELETE() - deletes all lines, which correspond to the criteria.

Related Functions:

DATABASE , CRITERIA , DBFIND , DBEXTRACT , DBSORT

1.84 DBEXTRACT([Cell])

DBEXTRACT([Cell])

Copies all data sets which correspond to the current criteria (corresponds to <Data-Extract>).

Cell determines the position to which the data shall be copied. Please make sure that there is enough space left around the copy-position, so that nothing will be overwritten by chance! If the parameter cell is omitted, inserting will be performed from current cursor-position onwards.

For details, refer to chapter four "database".

Example:

DBEXTRACT(A100) copies all matching datasets to A100.

Related Functions:

DATABASE , CRITERIA , DBFIND , DBDELETE , DBSORT

1.85 DBFIND([Cell])

DBFIND([Cell])

Searches for the first congruity of the database (corresponds to <Data-Find>).

Cell determines the start position of the search-process (A1 represents: from the beginning).

If the parameter cell is missing, the current cursor position serves as start position.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if searching was finished succesfully . If an error occured during the process, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see example below).

TurboCalc avoids to open a window, which indicates that the search has failed, within the macro mode. If you want to provide the user with this information, you have to program such a window in your macro by yourself.

For details, refer to chapter four "Database".

Example:

DBFIND(A1) seraches for the first column to match the current criteria.

DBFIND() resumes this process, after the first matching item has been found (TurboCalc stops searching after the first congruity).

A10 =BFIND(A1)

A11 =IFGOTO(#A10;A14)

A12 =MESSAGE("No matching items found !")

A13 =GOTO(A13)

A14 ... main routine ...

...

A30 =RETURN()

Related Functions:

DATABASE , CRITERIA , DBEXTRACT , DBDELETE , DBSORT

1.86 DBSORT(Ascending;[Cell])

DBSORT(Ascending;[Cell])

Sorts all data sets in ascending (or descending) order corresponding to the current column (corresponds to <Data-Sort Range>):

Ascending: Sorts the data-sets in ascending order, if this parameter is not 0 or omitted. The value nought or FALSE causes a sorting in descending order.

Cell determines the column according to which the sorting will be performed. If the parameter cell is missing, the sorting will be performed corresponding to the current cursor position. (If the column, which is used for the sorting process, lies outside a database range, the DBSORT command will be ignored!)

For details, refer to chapter four "Database".

Example:

DBSORT (A1) -sorts according to the first column (if the database range starts in column A).

Related Functions:

DATABASE , CRITERIA , DBFIND , DBEXTRACT , DBDELETE

1.87 Options

Options

DISPLAY(Title;Raster;Toolbar;Formulas;Zero)

FORMFEED(Flag)

LOCALE(NF1;NF2;DF;Currency;CPrefix;CSuffix;Inch)

PRINTFORMAT(LM;RM;UM;BM;STYLE;HEADLINE;HEADTEXT;FOOTER;FOOTTEXT;TITLE;RASTER)

PRINTRANGE(Activate;[Range])

REFRESH(Mode)

SHANGHAI(Mode)

SHEETFLAGS(Maxwidth;Maxheight;Calculation;Return;Direction;Icons)

SMARTREFRESH(Flag)

1.88 DISPLAY(Title;Raster;Toolbar;Formulas;Zero)

DISPLAY(Title;Raster;Toolbar;Formulas;Zero)

If called without parameter, this instruction corresponds to the menu item <Options - Display>, which allows the setting of the display-parameters of a TurboCalc-window.

Hint: All display-options only affect the current window - they can (and must) be set separately for every window!

Title: determines, if the column- and rowtitles (e.g. : A...ZZ; 1...200) shall be displayed.

Raster: determines, if a raster for visual cell separation shall be displayed.

Toolbar: determines, if the toolbar shall be displayed.

Formulas: This parameter affects the cell contents, if it is the result of a formula. It determines, whether the values or the formulas behind them shall be displayed.

Zero: determines, whether zero-values in cells shall be displayed or left out.

For each parameter you can specify the following options:

0 or FALSE: parameter not activated.

1 or TRUE: parameter activated.

2 toggle. Turns the status of a parameter into its opposite (activated becomes not activated and vice versa)

-1 or omission of parameter: The setting remains unchanged.

Any other value produces the error message "false parameter...".

Example:

DISPLAY(;;;2)

This instruction changes between displaying values or formulas as cell contents.

1.89 FORMFEED(Flag)

FORMFEED(Flag)

This function determines, whether a printed page should be finished with the control code Formfeed (12) or some linefeeds.

Normally, TurboCalc prints out several linefeeds to finish the page according to the page's length. This allows a free paper-format.

If you set FORMFEED to "on", you force TurboCalc to print out one single formfeed instead of the linefeeds. This requires to set the page-height directly in your printer-setups additionally to the setting within TurboCalc.

Flag: value 0 or FALSE means: no formfeed desired.

value 1 or TRUE means : formfeed desired.

Hint: Page-orientated printers (e.g. the ones, which use the laser- or inkjet technology) may require this option.

Attention: In spite of having activated this option, you have to determine the paper-format again within TurboCalc. The program needs this information to calculate the number of printable lines.

1.90 LOCALE(NF1;NF2;DF;Currency;CPrefix;CSuffix;Inch)

LOCALE(NF1;NF2;DF;Currency;CPrefix;CSuffix;Inch)

If you call this instruction without any parameter, it corresponds to the menu item <Options - Locale> and allows the determination of country-specific options.

If called with parameter you can change these options directly and thus avoid the respective selection-requester.

NF1 (NumericFormat1): determines the separator between decimals in front and after the decimal-point:

0: comma (1,23)

1: decimal point (1.23)

NF2 (NumericFormat2): determines the separator of the thousand figures in numbers.

0: comma (1,200)

1: decimal point (1.200)

2: apostrophe (1'200)

3: dash (1-200)

4: blank (1 200)

DF: determines the date format:

0: point (30.09.93)

1: dash (30-09-93)

2: slash (30/09/93)

Currency: determines the currency symbol:

0: DM

1: Dollar

2: user definition as follows below

3: Pound

4: Yen

5: FF (Franc)

6: ÖS (Schilling)

7: SFr (Franken)

CPrefix is the text, which shall be located in front of the amount of currency units.

CSuffix is the text, which shall be located behind the amount of currency units.

(One or both can be omitted as well. Please, replace them in the parameter notation by "").

Inch: TRUE (or 1) changes to "inch", FALSE (or 0) changes to "cm".

This flag is for the use in TurboCalc's requester only. The macro-instructions (e.g. PRINT) are not effected by this flag, there the values have to be entered in centimeters always (to ensure that the macros are compatible).

1.91 PRINTFORMAT(LM;RM;UM;BM;STYLE;HEADLINE;HEADTEXT;FOOTER;FOOTTEX

PRINTFORMAT(LM;RM;UM;BM;STYLE;HEADLINE;HEADTEXT;FOOTER;FOOTTEXT;TITLE;RASTER)

This instruction determines the outer appearance of the printout. If you call it without any parameters, it corresponds to menu item <Format-Print Format> .

LM: specifies the left margin in centimeters.

RM: specifies the right margin in centimeters.

UM: specifies the upper margin in centimeters.

BM: specifies the bottom margin in centimeters.

Hint: For compatiblity you have to enter the values in centimeters even if you have selected "inch" at <Options-Locale> . You may use "centimeter=inch/2.54" then (or even enter this formula as parameter).

Style: 0=Pica, 1=Elite, 2=Condensed

Headline: Determines, if the printout should receive a headline. TRUE or values unequal to zero means "Yes" while any other value suppresses the headline.

Headtext: is the text, which shall be printed as headline, if the parameter "Headline" is set to TRUE. The text has to be put in quotation marks.

You can also replace this parameter by a formula, which returns a text (e.g. LEFT...).

Footer: Determines, if the printout should receive a footer. TRUE or values unequal to zero means "Yes" while any other value suppresses the footer.

Foottext: is the text, which shall be printed as footer, if the parameter "Footer" is set to TRUE. The text has to be put in quotation marks.

You can also replace this parameter by a formula, which returns a text (e.g. LEFT...).

Title: Determines, if the line- or column titles shall be added to the print-out. TRUE or values unequal to zero means "Yes" while any other value suppresses the titles.

Raster: Determines if the print-out shall receive a raster for cell separation. TRUE or values unequal to zero mean "Yes" while any other value suppresses the raster.

Example:

PRINTFORMAT(2;2;2;2;0;TRUE;"Headline";FALSE;"Footer";FALSE;FALSE)

Determines the printformat as follows:

Margins 2 cm's on each side.

Style: Pica.

Headline on (text:"Headline").

Footer off (but text defined as:"Footer").

No titles nor raster.

PRINTFORMAT(;;;;;;;TRUE)

Fades in the footer and leaves the rest unchanged.

PRINTFORMAT(;;;;;;;;"page%P")

Changes the footer into the text "page" followed by the current page number.

1.92 PRINTRANGE(Activate;[Range])

PRINTRANGE(Activate;[Range])

Determines the range, which shall be transferred to the printer and corresponds to the menu item <Options - Print Range>

Activate: Determines, if the printrange-option shall be switched on or off. 0 or FALSE turns off the printrange and earmarks the entire sheet for print-out.

Range: Determines the range, which shall be transferred to the printer (if the parameter "Activate" has been set to a value unequal to zero). If this parameter is omitted, the current block will be marked.

Example:

PRINTRANGE() marks the current block as the new printrange.

PRINTRANGE(0) switches off a previously existing printrange.

PRINTRANGE(;A1:C5)

PRINTRANGE(TRUE;A1:C5) - both instructions determine block A1 to C5 as the new printrange.

1.93 REFRESH(Mode)

REFRESH(Mode)

Toggles between screen-refresh on or off, or forces a new refresh.

Mode:

0 (or FALSE) switches the screen refresh off.

1 (or TRUE) switches the screen refresh on.

The value 2 forces a new refresh. This is advisable, if the refresh has been switched off before and is now set to "on" again. TurboCalc does not execute a refresh automatically in this case, so you have to add this function manually.

Hint: This function turns off the screen refresh as well as the automatic formula calculation. This characteristic speeds up the macro execution, for it avoids the obligatory screen-reconstruction after every input (note hint number two). If you need the current cell results, a RECALC must be performed.

Hint: This instruction has its effect on macro- or ARexx instructions only. All normal inputs or instruction calls cause TurboCalc to refresh the screen immediately.

In spite of this fact the refresh should be switched on at the end of a every macro with REFRESH(TRUE). Then renew the screen with REFRESH(2) in order to get the current and correct cell contents.

Example:

Following macro extract accelerates the macro-execution:

A10 =REFRESH(0)

...here you may add the main routine to be accelerated.

A20 =REFRESH(1)

A21 =RECALC()

A22 =REFRESH(2)

A23 =RETURN()

1.94 SHANGHAI(Mode)

SHANGHAI(Mode)

(for OS2.0 and higher!)

Switches the so-called Shanghai-mode on or off:

Mode: Is the sum of the following two settings:

1: SHANGHAI (all new opened workbench-windows appear on the TurboCalc screen)

2: PUBSCREEN (determines, that the TurboCalc screen is put to front whenever a requester is going to appear on the TurboCalc screen.)

SHANGHAI(3) activates both options.

1.95 SHEETFLAGS(Maxwidth;Maxheight;Calculation;Return;Direction;Icons)

SHEETFLAGS(Maxwidth;Maxheight;Calculation;Return;Direction;Icons)

If you call this command with parameters, you can determine the individual behaviour of each sheet. If called without any parameter, it corresponds to menu item <Options - sheet>.

Maxwidth: determines the maximum width of the sheet (between 50 and 700 cells!).

Maxheight: Determines the maximum height of the sheet (between 50 and 9999999 cells!).

Calculation: Determines the kind of the recalculation process (automatic/manual). TRUE or values unequal to zero switches on the automatic recalculation after any change within the sheet. The value 0 or FALSE switches to manual recalculation.

Return: Determines, if the cursor shall be moved by pressing <Return>. TRUE or values unequal zero mean "Yes" while any other value suppresses the cursor-movement after <Return>.

Direction: Specifies the direction to which the cursor shall be moved: 0=down, 1=up, 2=left, 3=right.

Icons: Determines, if the saved sheet shall be provided with an icon. TRUE or values unequal zero means "Yes" while any other value suppresses the creation of icons.

1.96 SMARTREFRESH(Flag)

SMARTREFRESH(Flag)

Switches the so-called SmartRefresh on or off.

If it is activated, all windows will be saved temporarily in an intermediate memory as soon as they are overlaid by other windows or graphical objects. This procedure reduces the number of necessesary screen-reconstructions and speeds up the execution of TurboCalc accordingly.

The hook to this proceeding is the increased memory demand for the saving of the window contents.

Flag:

0: SmartRefresh is turned off completely.

1: SmartRefresh is enabled for sheet-windows.

2: SmartRefresh is enabled for chart-windows.

3: SmartRefresh is enabled for all windows.

If called without parameters, this instrcution corresponds to the menu-item <Options-Screen-SmartRefresh>, further details can be found there.

1.97 Menu Instructions

Menu Instructions

DELMENUITEM(Title;Item)

DELMENUTITLE(Title)

DELMENUSUB(Title;Item;Sub)

ADDMENUITEM(Name;Instruction;[Title;Item])

ADDMENUTITLE(Name;[Title])

ADDMENUSUB(Name;Instruction;[Title;Item;Sub])

NEWMENU()

SHOWMENU()

1.98 DELMENUITEM(Title;Item)

DELMENUITEM(Title;Item)

Deletes the menu item at the position title, item.

Title: Specifies the menu-title, which shall lose one menu-item (in numeric values: 0=first menu etc...).

Item: specifies the item, which shall be deleted (0=first...)

Example:

DELMENUITEM(0;1) deletes the menu item <Project - open...> in the standard menus.

1.99 DELMENUTITLE(Title)

DELMENUTITLE(Title)

Deletes the menu title at the position title.

Title: Specifies the menu-title number which shall be deleted. (0=start, 1=after first position...)

Example:

DELMENUITEM(0) deletes the entire menu title <Project> in the standard menus.

1.100 DELMENUSUB(Title;Item;Sub)

DELMENUSUB(Title;Item;Sub)

Deletes the menu item at the position title, item, sub.

Title: Specifies the menu-title, which shall lose one sub-menu-item (in numeric values: 0=first menu etc...).

Item: Specifies the menu-item, which shall lose one sub-menu-item (in numeric values: 0=first item etc...).

Sub: The sub item, which shall be deleted (0=first...)

Example:

DELMENUITEM(0;5;0) deletes the menu item <Project - import -Procalc> in the standard menus.

1.101 ADDMENUITEM(Name;Instruction;[Title;Item])

ADDMENUITEM(Name;Instruction;[Title;Item])

Inserts the new menu item name at the position title, item and assigns the instruction instruction to it.

Name: Is a string, which shall be displayed as menutitle-text.

Instruction:Is the instruction, which shall be executed through the selection of the respective menu item. Here, you might use any admitted macro- or ARexx instruction of TurboCalc.

Title:Is the title number to which the menu item shall be added (0=start, 1=after first position...). If title is missing, it will be set to the end of the title list automatically.

By replacing this parameter by a blank text ("") you specify this menu-item as "not-selectable".

Item:Is the menu-item number to which the menu item shall be added (0=start, 1=after first position...). If "item" is missing, it will be set to the end of the item list automatically.

For further information and a more detailed example, please refer to NEWMENU.

Example:

ADDMENUITEM("Macro 1";"MACROPLAY(Macro1)";1)

This instruction adds the menu item "Macro 1" to the end of the Edit-menu. If you select this menu item, a macro named "Macro 1" will be started immediately!

1.102 ADDMENUTITLE(Name;[Title])

ADDMENUTITLE(Name;[Title])

Inserts the new menu title name at the position title.

Name: Is a string, which shall be displayed as menutitle-text.

Title: Is the title number to which the menu item shall be added (0=start, 1=after first position...). If title is missing, it will be set to the end of the title list automatically.

For further information and a more detailed example, please refer to NEWMENU.

1.103 ADDMENUSUB(Name;Instruction;[Title;Item;Sub])

ADDMENUSUB(Name;Instruction;[Title;Item;Sub])

Inserts the new sub-menu item name at the position title, item, sub and assigns the instruction instruction to it.

Name: Is a string, which shall be displayed as sub-menu item text.

Instruction: is the instruction, which shall be executed through the selection of the respective sub-menu item. Here, you might use any admitted macro- or ARexx instruction of TurboCalc.

Title: Is the title number to which the sub-menu item shall be added (0=start, 1=after first position...). If title is missing, it will be set to the end of the title list automatically.

By replacing this parameter by a blank text ("") you specify this sub-menu item as "not-selectable".

Item: Is the menu-item number to which the sub-menu item shall be added (0=start, 1=after first position...). If item is missing, it will be set to the end of the item list automatically.

Sub: Is the sub-menu item number to which the submenu-item shall be added (0=start, 1=after first position...). If sub is missing, it will be set to the end of the sub-item list automatically.

For further information and a more detailed example, please refer to NEWMENU.

1.104 NEWMENU()

NEWMENU()

Deletes the current menu and allows the creation of a complete new menu with ADDMENUTITLE ADMENUITEM ADMENUSUB

Hint: Normally, menus are changed at once after a NEWMENU... or DELETEMENU... . This can be very time-consuming when you create a completely new menu. Therefore, the "Renewal" of the menu structure is switched off after a NEWMENU until the first SHOWMENU is executed. Therefore, SHOWMENU must always be the last instruction of a menu definition.

(If you only want to add or remove some menu-items subsequently, the use of SHOWMENU is not obligatory, except a NEWMENU has been called before).

Example:

NEWMENU

ADDMENUTITLE file

ADDMENUITEM open... LOAD

ADDMENUITEM "Save as..." SAVEAS

ADDMENUITEM END!!! QUIT

SHOWMENU

You find a further example in the file TurboCalc.STD2 - it contains the macro instructions for the standard TurboCalc-menu.

1.105 SHOWMENU()

SHOWMENU()

Shows the modified (resp. new) menu after the automatic display has been switched off by NEWMENU (see there).

This instruction should be located at the end of any menu declaration in script files.

1.106 Macro Control

Macro Control

BLOCKVARIABLE(Name;Block)

CLOSESHEET(Now)

DELETEVARIABLE(Name)

EXECUTE(File;Parameter;[Window])

MACROPLAY(Cell)

NEWSHEET(Name)

PRINT(Printer;File;NQL;Range;Page1;Page2;LPI;Colored;Break;Size;Width;Height)

QUIT([Flag])

RECALC([Mode])

RUN(File;Parameter;[Window])

SELECTSHEET(Name[;Windownumber])

START(Filename)

UNCHANGED()

VARIABLE(Name;Value)

1.107 BLOCKVARIABLE(Name;Block)

BLOCKVARIABLE(Name;Block)

Inserts a name in the variable list. The name can be subsequently used in formulas. For further details, refer to chapter five "Names".

Name: is a text to name the variable. (If possible, use simple name determinations without blanks or other specific characters).

Block: is the block, which shall be allocated to the name. If you omit this parameter, the current block will be used.

Note: If you want to allocate numerical or alphanumerical values to names, please use the VARIABLE instruction.

Example:

BLOCKVARIABLE("test";A1:C5)

allocates the block A1 to C5 to the name "test". This block can now be addressed with its reference A1:C5 or the name "test".

BLOCKVARIABLE("test")

the current block can now be addressed with the name "test".

1.108 CLOSESHEET(Now)

CLOSESHEET(Now)

Closes the current sheet (and asks for saving before quitting the sheet). If this instruction is called from the last opened sheet, TurboCalc will be terminated.

Now: determines, if TurboCalc should open a security-requester before quitting.

0: no further request desired.

1 or omission of the parameter:opens a requester and asks for saving before quitting the sheet.

Hint: Please note, that any other sheet can be declared as the "active" one after using this instruction. If you like to enter further commands, always use the SELECTSHEET instruction next.

1.109 DELETEVARIABLE(Name)

DELETEVARIABLE(Name)

Deletes the name specified by name in the name-list. For further details, please refer to the VARIABLE instruction and chapter five "Names".

Name: is a string, which specifies the name to delete.

Example:

DELETENAME("VAT")

Deletes the name VAT, but not the attached cell and its contents. If this variable is still used in a formula, you propably will get an error when recalculating.

1.110 EXECUTE(File;Parameter;[Window])

EXECUTE(File;Parameter;[Window])

Starts an external AmigaDOS-batch-file out of TurboCalc and waits until it is finished again. It corresponds to the RUN instruction but in this case the instruction is not only started but performed (OS2.0 and higher allows to use RUN only with a set script flag).

For further details and parameters, please refer to RUN.

Example:

EXECUTE("s:test")

Starts the file "s:test", which should consist of AmigaDOS-instructions.

1.111 MACROPLAY(Cell)

MACROPLAY(Cell)

Starts the macro-execution beginning with the cell given as parameter. If called without any parameters, this instruction corresponds to the menu-item <Macro-Play> and the appropriate selection-window will be opened.

Cell: is any cell information or name definition according to the TurboCalc standard. You can even refer to other sheets using the AT-function (or @).

If this instruction is executed in the ARexx-mode, ARexx is forced to wait until the macro is finished.

Attention: This instruction is not intended to call subroutines from within a macro. For this purpose use the command CALL.

1.112 NEWSHEET(Name)

NEWSHEET(Name)

Opens a new sheet and its corresponding window. The sheet receives the name specified in the parameter name or a default name, if the parameter is omitted.

Name: Is the string for the new sheet to name (this instruction does not replace the LOAD command. If you specify an already existing name (e.g. on diskette or harddisk), this file will not be loaded by NEWSHEET).

If the parameter name is omitted, the sheet will receive the name "Sheet1". TurboCalc asks then for the desired file-name before the first saving of the sheet.

1.113 PRINT(Printer;File;NQL;Range;Page1;Page2;LPI;Colored;Break;Size;Width;Heigl

PRINT(Printer;File;NQL;Range;Page1;Page2;LPI;Colored;Break;Size;Width;Height)

Printer: Determines, whether the output shall be sent to a printer (TRUE or not zero) or to a file (0 or FALSE).

File: is a text, which serves as the file name for the print-out redirection to a file.

NQL: Determines the print-out quality: TRUE or not zero switches to NearLetterQuality, the value 0 or FALSE prints the sheet in draft quality.

Range: TRUE (or not nought) sends the entire sheet to the printer, FALSE (or 0) only the following pages:

Page1, Page2: specifies the pages to print.

LPI: switches between 6 lpi (TRUE or not zero) or 8 lpi (FALSE or 0) (lpi= lines per inch).

Colored: toggles between color (TRUE or not zero) or black/white (FALSE or 0) print.

Break: Determines, if TurboCalc should pause after every page (FALSE or 0) or not (TRUE or not zero).

Size: Determines the paper size:

0: A4 normal (21*29.7)

1: A4 fanfold (21*72)

2: A3 normal (29.7*42)

3: user defined:

Width, Height: Determines the paper width and -height for the "user defined" page-dimensions, both in centimeters.

Hint: For compatiblity you have to enter the values in centimeters even if you have selected "inch" at <Options-Locale> . You may use "centimeter=inch/2.54" then (or even enter this formula as parameter).

Omitted parameters will not be changed.

If called without any parameters, this instruction corresponds to the menu-item <Project - Print> and the appropriate selection-window will be opened.

Further print parameters can be adjusted with the PRINTOPTIONS instruction.

Example:

PRINT(;)

Prints the current sheet as set before; no parameters are changed.

PRINT()

shows the window for the print-options selection (corresponds to <Project - Print...>)

PRINT(0;"ram:print.out")

Prints the current sheet to the file "ram:print.out" . the remaining parameters stay unchanged.

PRINT(0;"ram:print.out";;0;2;3)

dito, but limits the print-out to page 2 and 3.

1.114 QUIT([Flag])

QUIT([Flag])

This instruction ends a TurboCalc session.

If it is called within an ARexx script, this script receives the message "OK" (that means error value 0) first to terminate it normally.

You should not address TurboCalc via ARexx again after this instruction..

Flag: Determines, if a security-requester shall be opened before quitting:

0 or FALSE: no requester desired (even if unsaved sheets are still in memory).

1 or TRUE: requester desired, if unsaved sheets are still in memory.

If this parameter is not specified, TRUE will be assumed (corresponding to the menu item <Project - Quit>.

1.115 RECALC([Mode])

RECALC([Mode])

If you call this instruction without any parameter, you force TurboCalc to recalculate the entire sheet (corresponds to <Commands - Recalculate>). This is useful, if the automatic calculation is set to "off".

If the parameter mode is given, the automatic calculation can be switched on (TRUE or not null) or off (0 or FALSE).

Hint: The automatic calculation can be switched on or off with "SHEETOPTIONS", too (but this instruction requires further parameters). Because of the frequent use of the disabled automatic calculation (e.g. to speed up macros), this instruction has been added to the instruction set to simplify things.

Example:

RECALC()

Recalculates all formulas and actualizes the sheet.

RECALC(0)

Switches off the automatic calculation.

1.116 RUN(File;Parameter;[Window])

RUN(File;Parameter;[Window])

Starts an external program out of TurboCalc and waits until it is finished again.

File: is the name of the file to be started. Please, specify the full file path or set it before, so that the appropriate file can be found without problems.

Parameter: is the parameter-text, which shall be handed over to the program. If there are several parameters, simply separate them by blanks (as usual) and write them in one single string.

Window: specifies the name of the window, which shall be opened before starting the program. If this information is missing, the following circumstance will be assumed: "CON:0/0/0/640/200/TurboCalc/CLOSE/WAIT/AUTO". This command causes OS2.0 and higher to open a window only if it is actually needed. Furthermore, the window stays preserved until you touch the closing gadget.

(1.3 users receive the message "Please press return" after the execution. TurboCalc waits until this message is replied.)

Please make sure that the window specification is correct, otherwise the execution of this instruction will be denied.

Example:

RUN("dir";"dfO:") shows the directory of drive dfO:

Print-out in the background:

=PRINT(0;"ram:printer.out")

=RUN("RUN";">nil: copy ram:printer.out prt: QUIET")

This macro prints into the file "ram:printer.out" and starts the "copy" command in the background via RUN, which sends the file to the printer.

Thus, you are able to continue your work while printing.

1.117 SELECTSHEET(Name[;Windownumber])

SELECTSHEET(Name[;Windownumber])

This instruction selects a sheet as new Macro/ARexx-sheet. All further commands and cell references will only affect this sheet.

This instruction does not activate a certain window automatically; for this purpose use the ACTIVATEWINDOW instruction.

Name: specifies the name of the sheet (the name, which is displayed in the window title but without "-2..."). If the sheet-name ends in ".TCD", this tag can be omitted. Furthermore, all path information can be left out.

TurboCalc accepts the following names for the example file "DHO:TurboCalc/sheets/example.TCD":

"example", "example.TCD", "TurboCalc/example"...

Windownumber: specifies the number of the window (that means the "-XXX"- part of the window title). The numbering starts with 1. If the parameter is omitted or too big, the first window of this sheet will be taken.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if the selected sheet could be found. If the process has failed, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see similar example at DBFIND).

Hint: If you desire to read out the window name at a certain time, it is possible to use the function SHEETNAME (e.g. to buffer the current window temporarily).

Example:

SELECTSHEET("sheet1")

Selects the sheet, which is normally opened after the start-up phase.

SELECTSHEET("sheet1";2)

Selects a second split window of the same name as above produced with <Options-New Sheet Window>.

1.118 START(Filename)

START(Filename)

This instruction calls an external TurboCalc instruction-file.

It consists of macro instructions, ordered line by line (without quotation marks). The instructions can either be separated by brackets (macro style), or by blanks (ARexx convention).

The instructions GOTO, IFGOTO and CALL are not to be used in such a script file!

This is an optional method to write macro programs. It is suitable for longer programs (with the disadvantage that branching and loop constructions are not possible).

Hint: You can also use the specific sequence $$ as replacement for the parameter "filename", which causes TurboCalc to execute the internal script-file (which is run in the start-up phase). It creates the basic environment of TurboCalc (the menus and the screen). This file can be found on the origial disk under TurboCalc.STD2 and can be edited like any other TurboCalc sheet.

For further information, refer to chapter "Macro" "Executing a macro while starting TurboCalc".

Example:

START("ram:test")

with the following file "ram:test":

SELECT a1

PUT 3

SELECT A1:A10

SERIES(1;3)

1.119 UNCHANGED()

UNCHANGED()

This instruction marks the current sheet as "unchanged", which avoids the security request for saving the sheet when you try to close it or leave TurboCalc.

This is useful for macros that modify their sheets (e.g. to save intermediate calculation results). Thus, you can avoid the requester for saving the macro itself when quitting TurboCalc.

1.120 VARIABLE(Name;Value)

VARIABLE(Name;Value)

Inserts a name in the name list. The name can be subsequently used in formulas. For further details, refer to chapter five "Names".

Name: is a text to name the variable. (If possible, use simple name determinations without blanks or other specific characters).

Value: is the value, which shall be allocated to the name. It may be a number, a boolean value or a text.

Texts can also start with an equals sign "=", and thus introduce a formula or a reference/range, see examples below.

If called without any parameters, this instruction corresponds to the menu-item <Instructions - Define Names> and the appropriate selection-window will be opened.

Example:

VARIABLE("VAT";0.15)

Allocates the value 0,15 to the name VAT. This name can now be used as a constant in formulas, e.g. 200*(1+VAT). In case of a change of the cell "VAT" and a following <Commands - Recalculation> the formula will be changed as well.

VARIABLE("cell1";"=C5")

Cell C5 receives the name "cell1" and can now be addressed under this name in any allowed TurboCalc expression (e.g. in formulas).

VARIABLE("Test";"=CELL(-1;-1)*2")

Allocates a formula to the name "Test" (here: take the cell left above the calling cell and double its contents).

1.121 Screen Control

Screen Control

ACTIVATEWINDOW()

CHANGECOLOR(Color;Red;Green;Blue)

CHANGEWINDOW(X;Y;Width;Height)

ICONIFY()

MOVEWINDOW(X;Y)

OLDCOLORS()

SCREEN(Width;Height;Depth;Mode)

SETFONT(Characterset;Mode)

SHEETHIDE(Sheetname;Windownumber)

SHEETSHOW(Sheetname;Windownumber)

SIZEWINDOW(Width;Height)

STDCOLORS()

WINDOWTOBACK()

WINDOWTOFRONT()

1.122 ACTIVATEWINDOW()

ACTIVATEWINDOW()

Activates the current window.

1.123 CHANGECOLOR(Color;Red;Green;Blue)

CHANGECOLOR(Color;Red;Green;Blue)

This instruction allows to adjust the TurboCalc screen colors (or the workbench colors, if theTurboCalc windows are located on its screen):

Color: determines the color number which shall be set (0=backgroundcolor, 1=color 1...).

Red: determines the proportion of red in the color in a scaling from 0 to 15 (0= no red, 15= only red).

Green: determines the proportion of green in the color in a scaling from 0 to 15 (0= no green, 15= only green).

Blue: determines the proportion of blue in the color in a scaling from 0 to 15 (0= no blue, 15= only blue).

If you call this instruction without any parameter, a selection-window will be opened where the user may set the screen colors (see menu item <Options - Screen - Colors>).

Hint: If there are any doubts about the parameters, you can open the color requester for your adjustments (<Options - Screen - Colors>).

The color numbers in the requester start with 1, therefore you have to subtract 1, if you want to use the information as parameter - the red/green/blue-proportions can remain unchanged.

Example:

CHANGECOLOR(0;0;0;0)

sets the background color to black

CHANGECOLOR(1;15;0;0)

sets the character color to red, if you assume the normal color distribution.

1.124 CHANGEWINDOW(X;Y;Width;Height)

CHANGEWINDOW(X;Y;Width;Height)

Moves the current window on the screen and changes its size:

X,Y: specify the new coordinates.

Width, Height: determine the new dimensions of the window.

This is a combination of MOVEWINDOW and WINDOWSIZE.

1.125 ICONIFY()

ICONIFY()

This instruction corresponds to the menu item <Project - Iconify> and arranges that TurboCalc will be sent to a "waiting mode". All windows are closed; only a small "remember window" (or from OS2.0 on a respective icon) stays visible. By a click upon it, TurboCalc will be re-constructed, just like it was before the iconification.

1.126 MOVEWINDOW(X;Y)

MOVEWINDOW(X;Y)

Moves the current window on the screen:

X,Y: specify the new coordinates.

1.127 OLDCOLORS()

OLDCOLORS()

Sets the screen colors as they were determined when starting TurboCalc. This command is useful, if you have done (temporary) color manipulations (e.g. from within a macro) and you wish to return to the old settings.

1.128 SCREEN(Width;Height;Depth;Mode)

SCREEN(Width;Height;Depth;Mode)

Determines, if TurboCalc has to open an own screen, or if its windows appear on the workbench screen. Furthermore, this instruction also determines the outer appearance of the screen:

Width: determines the width of the screen (at least 600!). -1 or omission of the parameter corresponds to the width of workbench screen. The value 0 means: do not open an own screen, use workbench screen (all remaining parameters are not of importance, then.)

Height: determines the height of the screen (at least 200!). -1 or omission of the parameter corresponds to the height of the workbench screen.

Depth: determines the number of bitplanes, which is identical to the number of colors. -1 or omission of the parameter uses the workbench colors. 1 corresponds to two colors, 2= 4colors, 3= 8 colors and so on.

Mode: determines the screen mode in which the screen shall be opened (only for OS2.0 and higher). This parameter is only necessary for "unusual " formats. Regular formats are recognized by their screen dimensions.

This mode can either be entered as a number (screen-ID) or as a string. For further details, plaese refer to "Screen Modes" in the appendix.

If you call this instruction without any parameter, a selection-window will be opened where the user may set the screen mode. This corresponds to the menu item <Options - Screen-individual>

Example:

SCREEN()

opens a window to determine the screen.

SCREEN(;)

opens a screen with the dimensions and colors of the workbench.

SCREEN(0)

closes a previously opened screen and puts all windows onto the workbench.

SCREEN(640;256;3)

opens a 640*256 pixel screen with 8 colors.

SCREEN(640;470;2"VP")

opens a 640*480 pixel VGA-productivity screen.

1.129 SETFONT(Characterset;Mode)

SETFONT(Characterset;Mode)

This instruction sets the global character set (corresponding to the menu item <Options - Screen - Font>).

Character set: is the name of existing character set (format: name/size, e.g. "Times/13"). If this parameter is omitted, a selection window with a list of all available character sets will appear.

Mode: this parameter determines, which part of TurboCalc shall be presented in a new font.

0 (or omission of the parameter): selects the character set for all menus and all windows.

1: selects the character set only for the current window

2: selects the character set for the menus.

3: selects the standard character set of the current sheet (and corresponds to STDFONT)

Hint: You can only select the mode 0 via the TurboCalc menus. Different charactersets for different windows cannot be set via menu. Please, refer to the examples below.

1. Example:

=ADDMENUITEM("Menufont";"SETFONT("""";2)";5)

This macro instruction adds the menu item <Menufont> at the end of the menu <Options>. If you select this new item, you can determine a new character set for the menus.

2. Example:

=ADDMENUITEM WindowFont "SETFONT # 1"

(extracted from the TurboCalc.STD - start file)

Adds the menu item named "WindowFont" for selecting a new window-characterset.

1.130 SHEETHIDE(Sheetname;Windownumber)

SHEETHIDE(Sheetname;Windownumber)

Hides the specified resp. current window.

If you have only one opened window left, the call of this instruction will produce an error message. TurboCalc expects at least one open window before you can use SHEETHIDE (so open or create a new sheet, if necessary).

Name: specifies the name of the sheet (the name, which is displayed in the window title but without "-2..."). If the sheet-name ends in ".TCD", this tag can be omitted. Furthermore, all path information can be left out.

TurboCalc accepts the following names for the example file "DHO:TurboCalc/sheets/example.TCD":

"example", "example.TCD", "TurboCalc/example"...

Windownumber: specifies the number of the window (that means the "-XXX"- part of the window title). The numbering starts with 1. If the parameter is omitted or too big, the first window of this sheet will be taken.

If no parameter is given, the current window will be hidden. (This corresponds to <Options- Hide Sheet-Window> then).

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if the selected sheet could be found. If the process has failed, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see similar example at DBFIND).

Hint: This function can be used for a specific application: A macro adds a new menu item and hides itself after its execution. Thus, the macro-sheet does not disturb your normal proceeding but the new menu item can be called comfortably via menu or macro-requester.

1.131 SHEETSHOW(Sheetname;Windownumber)

SHEETSHOW(Sheetname;Windownumber)

Puts the specified sheet to front and activates it.

Furthermore, it redisplays the desired window, if it has been hidden before with the SHEETHIDE instruction.

Name: specifies the name of the sheet (the name, which is displayed in the window title but without "-2..."). If the sheet-name ends in ".TCD", this tag can be omitted. Furthermore, all path information can be left out.

TurboCalc accepts the following names for the example file "DHO:TurboCalc/sheets/example.TCD":

"example", "example.TCD", "TurboCalc/example"...

Windownumber: specifies the number of the window (that means the "-XXX"- part of the window title). The numbering starts with 1. If the parameter is omitted or too big, the first window of this sheet will be taken.

In the Macro Mode: If you use this instruction in the macro mode you have to note the following characteristic: the cell in which this instruction is located receives the value TRUE, if the selected sheet could be found. If the process has failed, the cell receives the value FALSE. This can subsequently be tested with IFGOTO (see similar example at DBFIND).

Attetion: The execution of this instruction does not declare the specified window for the current macro-window (the one, which is affected by all commands and macro-instructions).

For this purpose, please use SELECTSHEET.

If you call this instruction without any parameter, a selection-window will be opened . This corresponds to the menu item <Options - Show Sheet Window>.

1.132 SIZEWINDOW(Width;Height)

SIZEWINDOW(Width;Height)

Changes the size of the current window on the screen:

Width, Height: determine the new dimensions of the window.

1.133 STDCOLORS()

STDCOLORS()

Sets the standard colors of TurboCalc. This is useful, if you misplaced your colors before or for 1.3-users. For further details, please refer to the menu item <Options-Screen-StdColors>.

1.134 WINDOWTOBACK()

WINDOWTOBACK()

Puts the current window to the back of the screen.

1.135 WINDOWTOFRONT()

WINDOWTOFRONT()

Puts the current window to the front of the screen.

1.136 Instructions

Instructions

Many instructions presented in this chapter have, strictly speaking, no meaning for the macro programming.

The only reason why they have been included in the instruction set of TurboCalc is their application in the menu definition,

e.g. ADDMENUITEM ("Record Macro";"RECORD")

ABOUT()

DIASHOW()

HELP(Num;File)

NEWWINDOW()

POSWINDOW()

POSWINDOW2()

PROTECTFLAGS()

RECORD()

STOPRECORD()

SYSINFO()

1.137 ABOUT()

ABOUT()

Corresponds to the menu item <Project - About> and shows a window with information about the author and his program.

1.138 DIASHOW()

DIASHOW()

Corresponds to the menu item <Data - Show Chart...>.

1.139 HELP(Num;File)

HELP(Num;File)

Opens a text file (normally but not obligatory a help-file) and displays it:

Num: is the number of the help-text within the file (see below)

File: is the file to open. If this parameter is omitted, "TurboCalc.HELP" will be used instead.

The file is searched successively in the current directory, in the TurboCalc-directory (the one, in which TurboCalc itself is located) and in the S:-directory.

File structure: the help-file is a pure "ASCII"-text, where the lines are terminated with CR (that means <Return>, code 10).

Lines should not exceed more than 60 characters, the rest will be cut off.

To combine different "texts" in one file (to avaoid an unneccessary number of files), the double cross ("#") has been introduced as separator. It has to be the first character in a line, after the CR. These "texts" may be addressed with the parameter "num" (0 is the text until the first "#", then follows 1...).

Hint: This instruction can be used to show any text file. It is not limited to the help-file of TurboCalc only (see second example).

Example:

HELP(0)

shows the first help text.

HELP(0;"dfO:text")

shows the entire file "dfO:text" (until the first # at the beginning of a line, but this does not occur in texts normally!)

1.140 NEWWINDOW()

NEWWINDOW()

Corresponds to the menu item <Options - New Sheet-Window>.

1.141 POSWINDOW()

POSWINDOW()

Corresponds to <Options - Arrange Windows- show all>.

1.142 POSWINDOW2()

POSWINDOW2()

Corresponds to <Options - Arrange Windows- standard>.

1.143 PROTECTFLAGS()

PROTECTFLAGS()

Corresponds to the menu item <Options - Protection Flags>

1.144 RECORD()

RECORD()

Corresponds to the menu item <Macro - Start Record>.

This instruction should not be used in a macro for its only purpose is to facilitate the menu creation (call it after a leading ADDMENUITEM only).

1.145 STOPRECORD()

STOPRECORD()

Corresponds to the menu item <Macro - Stop Record>.

This instruction should not be used in a macro for its only purpose is to facilitate the menu creation.

1.146 SYSINFO()

SYSINFO()

Corresponds to the menu item <Help - Info> and displays the current system status.

1.147 Macro Control

Macro Control

This chapter contains the remaining "macro only" instructions. They allow the build-up of control constructions to manipulate the macro execution (conditional branching, loop constructions for (conditional) iterations and subroutines).

CALL(Cell)

GOTO(Cell)

IFGOTO(Condition;Cell)

LOOP()

MACRO(...)

RETURN([Cell])

STEP([Flag])

UNTIL(Condition)

WHILE(Condition)

1.148 CALL(Cell)

CALL(Cell)

Stops the current macro execution and branches to the position specified in "cell" where the macro will be continued.

TurboCalc saves the origin cell of the CALL instruction to resume the execution there after the first RETURN in the part the macro branched to before.

The instruction is similiar to GOTO, but is intended to introduce subroutines (GOTO does not save the last position and thus cannot return to its cell of call).

Cell: is an admitted cell reference (e.g. C10) or a cell name (e.g. loop), which specifies the position to resume the macro.

Hint: Normally, all cell references refer to the current sheet automatically (and not to the macro-sheet).

If you use this instruction, you intend to jump within the macro sheet, so all cell references refer to this one, ignoring the current sheet by the time of calling the instruction.

Therefore, the function CELL specifies the position relatively to the cell where the macro instruction is located (and not to the current cursor position!).

If you need to brach into another sheet, this can be realized with the use of the AT-reference (e.g. @DesiredMacro;C5).

1.149 GOTO(Cell)

GOTO(Cell)

Branches to the position specified in the parameter cell (unconditional jump). The next instruction will be taken from this "cell" and the macro will be continued there.

Cell: Is an admitted cell reference (e.g. C10) or a cell name (e.g. loop), which specifies the position to resume the macro.

Hint: Normally, all cell references refer to the current sheet automatically (and not to the macro-sheet).

If you use this instruction, you intend to jump within the macro sheet, so all cell references refer to this one, ignoring the current sheet by the time of calling the instruction.

Therefore, the function CELL specifies the position relatively to the cell where the macro instruction is located (and not to the current cursor position!).

If you need to brach into another sheet, this can be realized with the use of the AT-reference (e.g. @DesiredMacro;C5).

1.150 IFGOTO(Condition;Cell)

IFGOTO(Condition;Cell)

IFGOTO realizes the conditional branching. If the parameter "condition" is TRUE (or not null), the execution of the macro will be continued at the position given in "cell", otherwise in the next line.

Condition: any boolean value (with =,<>,<,<=,>,>=... also refer to "Formula - Booelans").

Cell: specifes the cell where the execution shall be continued, if the parameter "condition" is TRUE (for further details, see GOTO!).

GOTO is identical with the IFGOTO instruction and a TRUE condition.

Hint: Normally, all cell references refer to the current sheet automatically (and not to the macro-sheet).

If you use this instruction, you intend to jump within the macro sheet, so all cell references refer to this one, ignoring the current sheet by the time of calling the instruction.

Therefore, the function CELL specifies the position relatively to the cell where the macro instruction is located (and not to the current cursor position!).

If you need to brach into another sheet, this can be realized with the use of the AT-reference (e.g. @DesiredMacro;C5).

Example:

IFGOTO(TRUE;C5)

corresponds to GOTO(C5), for the condition is globally set to TRUE and cannot change.

The following program sequence executes the instructions in cell A1 to A13 ten times (10 is written to cell B1 and decremented by one successively (9,8,...). This proceeding is repeated while B1 is still bigger than 0.).

This allows a quick loop-construction (which is of inferior quality than the original construct LOOP. Please, refer to its explanation for further details).

Cell Macro-Instruction

A10 =PUT(10;B1)

A11

A12 =PUT(B1;CELLABS(B1;3))

A13

A14 =PUT(B1-1B1)

A15 =IFGOTO(B1>0;A11)

The instruction in cell A12 forces TurboCalc to set the cell with column 3 and row B1 (1 to 10) to value 1 to 10: cells B1 to B10 receive the contents 1 to10!

1.151 LOOP()

LOOP()

LOOP together with UNTIL or WHILE realizes a loop construct, that means that a part of your routine is passed several times, depeding on a certain condition: The instructions enclosed in LOOP and UNTIL or LOOP and WHILE are at least passed once. If the interpreter encounters the first WHILE and if the condition is TRUE (or, in case of UNTIL, FALSE), the part between LOOP and WHILE (resp. UNTIL) will be repeated until the stop-condition is FALSE (or, in case of UNTIL, TRUE).

The entire construction works without cell references (in contrast to IFGOTO), which improves legibility, error-safety, and reusability.

Please, use LOOP instead of IFGOTO whenever it is possible.

Hint: If you try to jump back within a loop by means of RETURN(), an error message will appear!

Example 1:

=PUT(1;A1)

= LOOP()

= PUT(A1+1;A1)

=WHILE(A1<10)

Example 2:

=PUT(1;A1)

=LOOP()

= PUT(A1+1;A1)

=UNTIL(A1=11)

Both examples write the start-value 1 to cell A1 and increment it by 1 as follows:

1. while it is smaller than 10.

2. until it is exactly 11.

(which leads, in this case, to an identical result)

Hint: as you might have noticed, it is possible to leave blanks between the instruction keyword and the equals sign.

Please, make use of this possibility to improve the structure (e.g. in complex loop constructions) and legibility of your routines.

Hint: you can use UNTIL and WHILE together as loop condition, but in most cases one single condition will be simpler and structurally clearer.

As mentioned before, it is possible to build up more complex loop constructions (e.g. loops within a loop).

Please, note the following Example:

=PUT(1;A1)

=PUT(1;A3)

=LOOP()

= PUT(1;A2)

= LOOP()

= PUT(A2+1;A2)

= PUT(A3+1;A3)

= UNTIL(A2=5)

= PUT(A1+1;A1)

=UNTIL(A1=5)

This example program counts cell A1 from 1 to 5 and (while doing this) cell A2 from 1 to 5. In every step A3 is increased by 1.

1.152 MACRO(...)

MACRO(...)

This instruction introduces a macro, for it allows to name your routine.

It is advisable to use this instruction as first command of every macro with its name set in the parameter brackets to improve the structure of the routine.

This command has no effect on the execution of a macro and its use is not obligatory!

The expression in brackets will be ignored - therefore you do not have to specify a valid formula or text in quotation marks.

Example:

MACRO(TestMacro)

1.153 RETURN([Cell])

RETURN([Cell])

This should be the last instruction of every macro for it terminates its execution.

If you have branched to a subroutine before (through CALL), it terminates the respective subroutine and returns to the first cell after the subroutine-CALL.

Cell: If you specify this parameter, the execution will be continued in this cell instead of the first cell after the subroutine-CALL.

1.154 STEP([Flag])

STEP([Flag])

Switches the single-step-mode on (if the parameter flag is omitted or not zero) or off (in case of 0 or FALSE).

This mode causes TurboCalc to display the next command to perform and to pause a while before its execution (tracing or trace mode).

It is very suitable for debugging purposes, please refer to the corresponding chapter seven "Macro-instructions" as well.

1.155 UNTIL(Condition)

UNTIL(Condition)

This is one stop-condition for the LOOP construct.

If the given condition results in FALSE, the program will be continued with the first command after LOOP (continuing the loop).

If the given condition results in TRUE, the superior WHILE-instruction will be skipped and the next command outside the loop will be executed (leaving the loop).

For further details, please refer to LOOP and WHILE.

1.156 WHILE(Condition)

WHILE(Condition)

This is the second stop-condition for the LOOP construct.

If the given condition results in TRUE, the program will be continued with the first command after LOOP (continuing the loop).

If the given condition results in FALSE, the superior WHILE-instruction will be skipped and the next command outside the loop will be executed (leaving the loop).

For further details, please refer to LOOP and UNTIL.

1.157 Special ARexx-Instructions

Special ARexx-Instructions

This chapter introduces the remaining Arexx instructions, which have no equivalent in the mutual macro/ARexx language. Remember, that all commands of chapter 12 except 12.13 and 12.14 can be used as macro and as ARexx instructions.

You should not call the following constructs from within a macro, because most of them serve to read out values from a certain sheet, which is in the macro language performed by stating the cell reference directly.

GETCURSORPOS

GETFORMULA [Cell]

GETVALUE [Value]

REM ...

1.158 GETCURSORPOS

GETCURSORPOS

This instruction reads out the current cell or -block of the selected sheet and saves it under RESULT in the regular reference format (A1 or C5:H7).

Hint: Make sure that you have inserted the "option results" instruction at the beginning of your ARexx script.

This internal ARexx-instruction allows the return of values.

Hint: this instruction can be used to buffer the cursor position temporarily.

Example:

/*read out old position*/

GETCURSORPOS

oldpos = RESULT

/* Here you may add your routine */

.....

/* set old position */

"SELECT" oldpos

1.159 GETFORMULA [Cell]

GETFORMULA [Cell]

This instructions is used to read out TurboCalc formulas and to hand them over to the ARexx script. There, they can be adressed via the RESULT variable.

Cell: can be any admitted cell reference. If this information is missing, the formula of the current cell will be read out and displayed.

If the current cell does not contain a formula, the blank text "" will be returned.

Hint: Make sure that you have inserted the "option results" instruction at the beginning of your ARexx script.

This internal ARexx-instruction allows the return of values.

Example:

PUT "=A1+A2" A3

GETFORMULA A3

SAY RESULT

reads out the formula of cell A3 (which has been defined as =A1+A2 before)

RESULT is filled with =A1+A2

"SELECT A3"

GETFORMULA

SAY RESULT

dito, but the SELECT statement declares cell A3 as current one, so you can leave out the cell reference in the GETFORMULA instruction.

1.160 GETVALUE [Value]

GETVALUE [Value]

This instructions is used to read out TurboCalc values (numbers, texts...) and to hand them over to the ARexx script. There, they can be adressed via the RESULT variable.

Value: can be any admitted formula (a simple numerical value, cell reference, functions...), except pure strings (alphanumercial data).

If this information is missing, the contents of the current cell will be read out and shown.

Hint: normally, this instruction will be used only to read out the contents of a cell (calculations can be executed directly out of ARexx). Please, check the first two examples.

Hint: Make sure that you have inserted the "option results" instruction at the beginning of your ARexx script.

This internal ARexx-instruction allows the return of values.

Example:

GETVALUE A1

SAY RESULT

reads out the value of cell A1 and puts it to the screen.

"SELECT A1"

GETVALUE

SAY RESULT

dito, but here GETVALUE (without parameter) reads out the current cell.

GETVALUE "5+5"

returns 10 to the ARexx variable RESULT.

GETVALUE "SUM(A1:A10)"

calculates the sum of cell A1 to A10 and transfers it to RESULT (please note the combination of ARexx and TurboCalc macro instructions).

1.161 REM ...

REM ...

(also ** or -)

This instruction has no operational effect. The text you may state behind the key-word, is treated as a pure remark.

This is applicable for macro-script files only (please, refer to the START command). Within macros it is possible to write any remark as normal text.

1.162 Table of Contents

Table of Contents

LASTROW()

LEFT(Num

RIGHT(Num

UP(Num

DOWN(Num)

LINE(Line)

FIND(Text;Part;Case;Columns;Range)

SELECT([Block])

Input Instructions

BEEP()

DELAY(Time)

INPUT(Text[;Title];[Cell])

INSERTFORMULA()

INSERTMACRO()

INSERTNAME()

MESSAGE(Text[;Title])

PUT(Contents[;Cell])

REQUEST(Text[;Title])

Load / Save

CSVINSERT([Block];[Name];[Separator])

CSVLOAD([Name]);[Separator])

CSVSAVE([Name]);[Separator])

CSVSAVEBLOCK([Block];[Name];[Separator])

LOAD([Name])

LOADCONFIG()

PROCALCINSERT([Block];[Name])

PROCALCLOAD([Name])

SAVE([Name])

SAVEAS([Name])

SAVEBLOCK([Block];[Name])

SAVECONFIG()

SYLKINSERT([Block];[Name])

SYLKLOAD([Name])

SYLKSAVE([Name])

SYLKSAVEBLOCK([Block];[Name])

TCDINSERT([Block];[Name])

Database Instructions

CRITERIA([Range])

DATABASE([Range])

DBDELETE()

DBEXTRACT([Cell])

DBFIND([Cell])

DBSORT(Ascending;[Cell])

Options

DISPLAY(Title;Raster;Toolbar;Formulas;Zero)

FORMFEED(Flag)

LOCALE(NF1;NF2;DF;Currency;CPrefix;CSuffix;Inch)

PRINTFORMAT(LM;RM;UM;BM;STYLE;HEADLINE;HEADTEXT;FOOTER;FOOTTEXT;TITLE;RASTER)

PRINTRANGE(Activate;[Range])

REFRESH(Mode)

SHANGHAI(Mode)

SHEETFLAGS(Maxwidth;Maxheight;Calculation;Return;Direction;Icons)

SMARTREFRESH(Flag)

Menu Instructions

DELMENUITEM(Title;Item)

DELMENUTITLE(Title)

DELMENUSUB(Title;Item;Sub)

ADDMENUITEM(Name;Instruction;[Title;Item])

ADDMENUTITLE(Name;[Title])

ADDMENUSUB(Name;Instruction;[Title;Item;Sub])

NEWMENU()

SHOWMENU()

Macro Control

BLOCKVARIABLE(Name;Block)

CLOSESHEET(Now)

DELETEVARIABLE(Name)

EXECUTE(File;Parameter;[Window])

MACROPLAY(Cell)

NEWSHEET(Name)

PRINT(Printer;File;NQL;Range;Page1;Page2;LPI;Colored;Break;Size;Width;Height)

QUIT([Flag])

RECALC([Mode])

RUN(File;Parameter;[Window])

SELECTSHEET(Name[;Windownumber])

START(Filename)

UNCHANGED()

VARIABLE(Name;Value)

Screen Control

ACTIVATEWINDOW()
CHANGECOLOR(Color;Red;Green;Blue)
CHANGEWINDOW(X;Y;Width;Height)
ICONIFY()
MOVEWINDOW(X;Y)
OLDCOLORS()
SCREEN(Width;Height;Depth;Mode)
SETFONT(Characterset;Mode)
SHEETHIDE(Sheetname;Windownumber)
SHEETSHOW(Sheetname;Windownumber)
SIZEWINDOW(Width;Height)
STDCOLORS()
WINDOWTOBACK()
WINDOWTOFRONT()
Instructions
ABOUT()
DIASHOW()
HELP(Num;File)
NEWWINDOW()
POSWINDOW()
POSWINDOW2()
PROTECTFLAGS()
RECORD()
STOPRECORD()
SYSINFO()
Macro Control
CALL(Cell)
GOTO(Cell)
IFGOTO(Condition;Cell)
LOOP()
MACRO(...)
RETURN([Cell])
STEP([Flag])
UNTIL(Condition)
WHILE(Condition)
Special ARexx-Instructions
GETCURSORPOS
GETFORMULA [Cell]
GETVALUE [Value]
REM ...

1.163 Index

DBDELETE()

DBEXTRACT([Cell])

DBFIND([Cell])

DBSORT(Ascending;[Cell])

DELAY(Time)

DELETEVARIABLE(Name)

DELMENUITEM(Title;Item)

DELMENUSUB(Title;Item;Sub)

DELMENUTITLE(Title)

DIASHOW()

DISPLAY(Title;Raster;Toolbar;Formulas;Zero)

DOWN(Num)

E

EXECUTE(File;Parameter;[Window])

F

FILL(Mode;[Block])

FIND(Text;Part;Case;Columns;Range)

FONT([Num];(Characterset];[Block])

Formatting Instructions

FORMFEED(Flag)

FRAME(Left;Right;Top;Bottom;[Block])

FREEZE(Cell)

G

Generals

GETCURSORPOS

GETFORMULA [Cell]

GETVALUE [Value]

GOTO(Cell)

GOTOCOLUMN(Column)

GOTOLINE(Line)

H

HELP(Num;File)

HIDE(Row;[Block])

I

ICONIFY()

IFGOTO(Condition;Cell)

Input Instructions

INPUT(Text[;Title];[Cell])

INSERTFORMULA()

POSWINDOW2()

PRINT(Printer;File;NQL;Range;Page1;Page2;LPI;Colored;Break;Size;Width;Height)

PRINTFORMAT(LM;RM;UM;BM;STYLE;HEADLINE;HEADTEXT;FOOTER;FOOTTEXT;TITLE;RASTER)

PRINTRANGE(Activate;[Range])

PROCALCINSERT([Block];[Name])

PROCALCLOAD([Name])

PROTECTFLAGS()

PROTECTION([Write];[Formula];[Block])

PUT(Contents[;Cell])

Q

QUIT([Flag])

R

RECALC([Mode])

RECORD()

REFRESH(Mode)

REM ...

REMOVE(Data;[Block])

REQUEST(Text[;Title])

RETURN([Cell])

RIGHT(Num

ROWHEIGHT(Height;Block)

RUN(File;Parameter;[Window])

S

SAVE([Name])

SAVEAS([Name])

SAVEBLOCK([Block];[Name])

SAVECONFIG()

Screen Control

SCREEN(Width;Height;Depth;Mode)

SELECT([Block])

SELECTSHEET(Name[;Windownumber])

SERIES(Type;Increment;Columns;Range)

SETFONT(Characterset;Mode)

SHANGHAI(Mode)

SHEETFLAGS(Maxwidth;Maxheight;Calculation;Return;Direction;Icons)

SHEETHIDE(Sheetname;Windownumber)

SHEETSHOW(Sheetname;Windownumber)

SHOW(Row;[Block])

SHOWMENU()