AmiCAD

FLORAC Roland

COLLABORATORS			
	TITLE : AmiCAD		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	FLORAC Roland	August 26, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmiCAD

1.1 Sommaire

```
                        ---------------------------------
  AmiCAD Version 1.3  June 4 1998
       ---------------------------------

Warning: this guide is an incomplete translation of the
french guide.
Only some items are translated for the moment.
```

1.2 Distribution

AmiCAD has been written by R.Florac. The AmiCAD package is
Copyright © 1998 R.Florac, it's not public domain.

AmiCAD is distributed under the concept of giftware/emailware.
It can be used only for personal usage, professional usage is stricly
forbidden without my permission, contact-me for that.

It is allowed to put this program only
- into AmiNet
- on AmiNetCDs

You can reach me sending me your E-mails to the address
Roland.Florac@wanadoo.fr

1.3 Requirements

AmiCAD is a program written in C (SAS C Compiler 6.58), and also with
Devpac 3.14 for some little assembly routines.

AmiCAD requires the following:
- Workbench 3.0+
- 68020 processor or better.

It's an electronics vector sheets editor. It uses only integers for
faster and simpler working. His ARexx interface gives it more than 130
commands, user defined functions are also useable. It can use his own
variables (integer maths and strings).
The
bgui.library
version 39+ can be used for loading and
looking for symbols, allthough it's not necessary for running the program.
This library is not included in the package, you can get it on Aminet.
The program can perfectly work without it, but the loading of libraries will
be a lot harder, and the choice of the components so...

The program can work with many windows in the same time, while the memory

```
  isn't filled. The advanced options of the 3.0 system are used:
              AppIcon
              ,

              Pools
              ...
```

1.4 Installation

```
                The simplest method of installation  is  to  use  the  provided ↩
                    Installer
script. Installer program is required for this to work.
The preferences files AmiCAD.prefs and AmiCAD.keys must to be copied in the
same drawer than the AmiCAD program.
The AmiCAD.guide
              help
               file can be copied everywhere
you like, you only have to set the AmiCAD tooltype
              HELPFILE
               to the
corresponding path (HELP: for example).

  WARNING: the program can't be runned without the 3.0 ROM system (or 3.1).
```

1.5 Running the program / ToolTypes

```
                The stack used by AmiCAD can be set to 4 kB only.
AmiCAD can be started from the CLI or the Workbench (by double
  clicking on it's icon).

  Running from a CLI (or Shell)
    The template of the command is:
  AmiCAD FILE/M,HELPFILE/K,LIBS/K,CLIPS/K,STARTUP/K,MACRO/K,SHEET_WIDTH/N, ↩
     SHEET_HEIGHT/N
    The program have not to be "runned", it detach itself from the
console.
    AmigaDOS jokers can be used:
    AmiCAD #?.sheet   ==> all the files with the extension ".sheet"
will be loaded in a different window.
    You can start the program without giving it a file name: the
title "NoName" will be used by default.

    The keyword HELPFILE gives the path where the help file can be found:
    Example: AmiCAD HELPFILE HELP:AmiCAD.guide

    The keyword LIBS gives the path where are stored the symbol libraries:
    Example: AmiCAD LIBS Work:AmiCAD/Symboles

    The keyword CLIPS gives the path where are stored the clips:
    Example: AmiCAD CLIPS Work:AmiCAD/Clips

    The keyword STARTUP gives the program the name of an ARexx script
```

that will be runned at startup.
Example: AmiCAD STARTUP startup.AmiCAD

The keyword MACRO gives a command that will be executed at startup.
Example: AmiCAD MACRO=LOADPREF("Config2")

The keyword gives the width of the SuperBitmap window.
Example: AmiCAD SHEET_WIDTH=1000

The keyword SHEET_HEIGHT gives the height of the SuperBitmap window.
Example: AmiCAD SHEET_HEIGHT=700

All this keywords can be associated in a single command:
AmiCAD HELPFILE HELP:AmiCAD.guide LIBS Work:AmiCAD/Symbols NewSheet

Running from Workbench
You only have to click twice on the icon of the program or on an icon associated to a file saved with AmiCAD.

The program icon accepts the ten tooltypes described there:

WINDOW
,
HELPFILE
,
STARTUP
,
LIBS
,
MACRO
,
CLIPS
,
X_ICON
,
Y_ICON
,
SHEET_WIDTH
,
SHEET_HEIGHT
,
GRIDSIZE
.

Use the Workbench Information menu to edit and modify them.

When the program is running, all the requesters that are opened can be closed by clicking on the gadgets or with the keyboard (sometimes with the

right mouse button also). ESC can replace a click on CANCEL, ENTER can replace YES or OK.

1.6 Title bar format

There are some indications on the title bar of the windows:
- le name of the file associated to the window (with the complete path or not),
- some letters beetween two braces []:,
  - a + sign if the document has been modified
  - the letter G if snap to grid is valid,
  - the letter R if the components are placed with their Reference,
  - the letter V if the components are placed with their Value,
  - the letter N if the components are placed with their pins numbers,
  - the letter L if lines are pulled while moving components,,
  - a number that gives the width of the actual lines that are drawn (0 for dashed lines).
- the cursor coordinates X=...,Y=...

The windows have two proportional gadgets to scroll big sheets, larger than the screen. The windows can be iconified, then they have no more gadgets, but they take only a little place on the screen. Select this window and click on the right button of the mouse to open it again.

1.7 The HELPFILE tooltype

This tooltype gives to the program the path to find the guide file. This word must be followed by the sign '=' and the path and name of the guide file. So you can place it where you want and even rename it.

Example:

```
HELPFILE=HELP:AmiCAD.guide
```

1.8 The WINDOW tooltype

This tooltype gives the program the dimensions and emplacement
of the window that is opened when the program is runned. The
word must be followed by the sign '=', then the coordinates of
the upper left corner, then the window width and height. No space
can be inserted beetween the fields, just a ",".

Note: if you load a file at startup, this tooltype will be
ignored, the datas will be taken into the file.

Example:
    WINDOW=0,1,1000,500   If the screen is smaller than the
          requested dimensions, the window is
          opened at the maximum available.

1.9 The STARTUP tooltype

This tooltype gives to the program the name of an
ARexx script
that will be runned at startup. So you can
define
the usual functions you often use,
if you want.

Examples:
    STARTUP=startup.amiCAD
    STARTUP=startup

1.10 The LIBS tooltype

This tooltype gives the path where the symbols files are found.
The default path is the Libs directory, in the AmiCAD directory.

Examples:
    LIBS=Work:AmiCAD/Symbols
    LIBS=Symbols

See also:
    LIBSPATH

1.11 The CLIPS tooltype

Ce type d'outil permet de spécifier l'emplacement (le chemin)
des clips définis par l'utilisateur. Par défaut, ce chemin est
celui du répertoire nommé "Clips", situé dans le chemin courant
du programme.

Exemples:
    CLIPS=Work:AmiCAD/Clips
    CLIPS=Symboles_utilisateur

See also :
CLIPPATH

1.12 The MACRO tooltype

This tooltype can specify a command that will be executed at the program startup. All ARexx commands can be used, you can also specify many commands, using the : separator.

Example:
MACRO=LOADPREF("Configuration2"):LOADLIB("Symboles spéciaux")

1.13 The X_ICON tooltype

This tooltype is used to fix the icon associated to AmiCAD on the Workbench screen.

Example:
X_ICON=10 puts the icon at the left of the screen.

See also:
Y_ICON

1.14 The Y_ICON tooltype

This tooltype is used to fix the icon associated to AmiCAD on the Workbench screen.
Example:
Y_ICON=500 puts the icon at the bottom of the screen.

See also:
X_ICON

1.15 The SHEET_WIDTH tooltype

This tooltype defines the width of the document that will be opened when the program is runned (default value). The window is a SuperBitmap window, all the objects will be drawn in it. Big values require a large space of chip memory, but you can then draw big sheets on the same document.
Adjust this value to your needs and to your configuration.

This value will be adjusted to a multiple of 16.

Example: SHEET_WIDTH=750

See also:
SHEET_HEIGHT

1.16 The SHEET_HEIGHT tooltype

This tooltype defines the heigth of the document that will be opened when the program is runned.

See also:
SHEET_WIDTH

1.17 The GRIDSIZE tooltype

This tooltype defines the grid size. This default size is 10.
If you prefer using a size of 5, use the example below:

Example: GRIDSIZE=5

1.18 AmiCAD menus

AmiCAD handles 5 menus, they can be used with the right mouse button, like any other application on Amiga. Many of them can be called using the keyboard, without the right Amiga key.

Project

Drawing

Edit

Macros

Preferences
Many of these menus can be called using the ARexx function MENU
.

1.19 Project menu

This menu has 16 entries:

Load file
Load a file in the active window.

Save file

Save the current document.

Save as
Sve the current document using the asl requester.

Save IFF
Save the current document in an IFF file.

Rename
Rename the current document.

Filenote
Annotate a file.

Kill file
Delete a file.

Iconify
Iconify the current window.

Hide
Close the current window, the document is allways in memory.

Other window
Open a new window.

Initialize
The current window is cleaned.

Print
Printing of the current document.

Informations
Displays some informations.

Sheets
Displays the document list.

Help~
Displays the AmigaGuide help.

Quit
Close all the documents.

1.20 Project menu/Load file

You can choose the document to load with the asl.library requester.
This library is loaded only when needed. The window takes the name of the loaded file.

If the window was not empty and his content had not been saved, a request will ask you for saving the document.

Keyboard shortcut : AMIGA-O (open) or O

```
  Note: you can use the
              AppIcon
               present on the Workbench screen for
  loading a file.

See also:
              LOAD
              ,
              OPEN
              .

ARexx call: MENU("Charger")
```

1.21 Project menu/Save file

```
                  The document is saved using the current file name of the  ↩
                     window.

Keyboard shortcut : AMIGA-S (save) or S

See also:
              SAVE
              .

ARexx call: MENU("Save~file")       Warning: solid space beetween
             (ALT SPACE) the words save and file.
```

1.22 Project menu/Save as

```
                  The current document is saved using the asl.library requester  ↩
                     to
  choose the file.

  If the file allready exists a requester will ask you to confirm
  if you want to overwrite it.

Keyboard shortcut : AMIGA-A (save As) or A

See also:
              SAVE
              .

ARexx call: MENU("Save~as")     Warning: solid space beetween
        (ALT SPACE) the words save and as.
```

1.23 Project menu/Save IFF

The current document is saved in a file using the IFF format.
You can then load this file in a bitmap drawing program
like Personal Paint, Deluxe Paint or other... You can also
import the file with a program like ProPage or Wordworth.
The saving is done with only two colors (black lines, white
background). The grid is not represented.

The file is selected using the asl.library requester.

There is no keyboard shortcut for this entry.

See also:
SAVEIFF
.

ARexx call: MENU("Save~IFF") Warning: solid space beetween
(ALT SPACE) the words save and IFF.

1.24 Project menu/Rename

The asl.library requester is opened and you can choose a file,
the current window is renamed using this file name.

Keyboard shortcut : AMIGA-= or =

See also:
FILENAME
.

ARexx call: MENU("Rename")

1.25 Project menu/Filenote

The asl.library requester is opened, if you select a file name
a requester asks you for a note to attach to the file.

This note will be displayed using the AmigaDOS command List or
with the Workbench menu Icon/Information.

There is no keyboard shortcut for this entry.

ARexx call: MENU("Filenote")

1.26 Project menu/Kill file

The asl.library requester is opened, if you select a file name, it
will be deleted. Warning: you can't recover the file...

There is no keyboard shortcut for this entry (dangerous...).

ARexx call: MENU("Kill~file") Warning: solid space beetween
(ALT SPACE) the words kill and file.

1.27 Project menu/Iconify

The window is closed, a small window is opened in the top of the screen, including only the name of the document (without the path).
You can select this window and use the right button to reopen it.
This menu is usefull when you work with multiple windows.

You can place this window whenever you want, when this menu will be used again, it will return to this place.

To close an iconified window without opening it, type CTRL-Q with the keyboard when it's selected.

Keyboard shortcut : AMIGA-I (icône) I

ARexx call: MENU("Iconify")

1.28 Project menu/Hide

The window is closed but the document remains in memory. To reopen this window, click on the appicon (on the Workbench screen) if it's the only window. If you are working with another window, use a double click on the right button of the mouse: a requester is opened with a button for each document that is in memory, just click on the button of the document you want to work with.

Keyboard shortcut : AMIGA-$ or $

ARexx call: MENU("Hide")

1.29 Project menu/Other window

A new window is opened, the asl.library requester is opened ↩
and
you can select a file to load in it.
If you want to open a window without loading a file, just use the F4 key.

Keyboard shortcut : F3 or F4 (no asl.requester)

The keyboard shortcuts can be used selecting an iconified window and using the key.

See also:
NEW
,

OPEN
.

ARexx call: MENU("Other~window") Warning: solid space beetween
(ALT SPACE) the words Other and window

1.30 Project menu/Initialize

All the objects of the current window are deleted. If the document was modified but not saved, a requester is opened asking you to confirm the operation.

There is no keyboard shortcut for this entry.

ARexx call: MENU("Initialize")

1.31 Project menu/Print

The document is printed using the printer.device. The current system Preferences are used (printer driver...)

A requester will ask you for a ratio, if you reply by a value of 1 a pixel at screen will be printed like a dot on the paper, if you reply by 2, the document will be printed twice larger. Try to find the best value for your printer density and the dimensions of your document.
WARNING: if the value of the ratio is 0, the operation is aborted. Large values consume many chip memory.

If you reply with a good positive value, a second requester will ask you if you want to print the document with a rotation or not, this feature lets you print the document horizontally (no rotation) or vertically.

The windows are iconified during this process to have more chip memory free. If you got any problem, try to close some windows or screens not used at this moment.

Keyboard shortcut : AMIGA-P (Print) or P

See also:
PRINT
ARexx call: MENU("Print")

1.32 Project menu/Informations

A requester is opened to display some d'informations : copyright, number of objects in the current document, name of the ARexx
port
, free memory...

Keyboard shortcut : AMIGA-K or K

ARexx call: MENU("Informations")

1.33 Project menu/Sheets

A requester is opened with all the document names and the number of objects they include. At the left of the display, a number is displayed. This number is the number of the window. It's usefull because you can select any window without the mouse but with the keyboard, simply by pressing a ALT key with one of the numeric keys (0 to 9): then the window with this number is placed in front of the screen.
So the keyboard shortcut ALT-1 places the first window in front of the screen, ALT-2 puts the second window, and so on. You can also use the - and + keys (allways with a ALT key) to go to the previous or to the next window.
The sign (+ or -) that is present at the right of the window number, in the requester says if the document has been modified (+) or not (-).

Another solution to call a window is to use a double click on the right mouse button: a requester will be opened with a button for each document. Just click on the right button to put his window in front of the screen.

There is no keyboard shortcut for this entry.

See also:
REQSHEET
.

ARexx call: MENU("Sheets").

1.34 Project menu/Help

A requester is opened to ask you the name of the node of the AmiCAD.guide file you want to be displayed. This file contains many items, there is a node for each ARexx macro, with his name (Ex: COPY, PASTE...)
You can also have a help for each menu entry selecting this menu entry with the right mouse button and pressing the HELP key.
When there is an error displayed in a requester, if this error is relative to an ARexx function, just type the HELP key while the requester is opened, the AmiCAD.guide will be opened to the corresponding node.

Keyboard shortcut: AMIGA-? or ?

See also:
HELP
.

ARexx call: MENU("Help")

1.35 Project menu/Quit

All the documents present in memory are closed. If some of them was
modified and not saved, a requester will ask you to continue or not.
If all is closed, the program frees all the memory he was using.

Keyboard shortcut : AMIGA-Q (all the windows are closed)
Q (the current window only is closed)

See also:
CLOSE
.

Arexx call: MENU("Quit") Note: no saving will be proposed.

1.36 AmiCAD AppIcon

When AmiCAD is
running
, an AppIcon is placed on the Workbench
screen. His name is AmiCAD. Its purpose is to load any document
dragging its own icon on this AppIcon. Simple, no?

This AppIcon has another function: clicking twice on it puts the AmiCAD
screen at frontend. You can use it to recover a window that was closed
previously (see menu
Project/Hide
).

If the program seems locked after executing an ARexx script you can
also try to click on it, if this situation is caused by a bad use
of the ARexx function
LOCK
, the problem will be solved.

1.37 Memory handler

Pools are handled by the operating system version 3.0 and more.
Its purpose is to have a better memory handler, it's fastest
and it reduces memory fragmentation.

1.38 Drawing menu

This menu is for drawing objects on the document. All these objects
are vector oriented. They can be modified with a double click from
the left mouse button (a requester is opened with gadgets, depending
on the object type).

This menu has 26 entries:

Grid size

Snap on grid

Choose component

Place component

Place reference

Place value

Place pins numbers

Rotation

Reflection

Alternate symbol

Normal position

Place line

Orthogonal mode

Continuous~drawing

Double~line

Bus

Dashed~line

Any~width

Place box

Place ellipse

Place arc

Place junction

Place text

Place input connector

```
              Place output connector

              Redraw all
```

1.39 Drawing menu/Grid size

```
                With this menu, you can define the size of the grid that is
  used to place the objects on the document. The default size is 10.
The components defined in the symbols libraries all use this
  size. But you can choose another size if you want, some users
  prefer working with a size of 5.

Keyboard shortcut: AMIGA-$\mathrm{\mu}$ or $\mathrm{\mu}$

See also:
              SETGRID
              .

ARexx call: MENU("Grid size")   Warning: solid space beetween
   or MENU("Grid")        (ALT SPACE) the words Grid and size.
```

1.40 Drawing menu/Snap on grid

```
When this entry is marked, the objects are placed on the grid,
  if it's not marked you can place the objects whenever you want, but if
  you place some components like that, you will get some difficulties to
  place the wires...

Keyboard shortcut: AMIGA-G or G

ARexx call: MENU("Snap")
```

1.41 Drawing menu/Choose component

```
                A requester is opened to look for a component in the libraries.
  This function needs the
              bgui.library
              .
The window has two lists: the left one is for the libraries,
  the right one for the symbols.

You can load or flush any library using the buttons. The buttons
  labeled "Before" and "After" are used to change the order of the
  libraries in the list, symbols are searched in the first library,
  then in the second...
The other buttons have the same role that the equivalent menus.

The selected component is displayed in the upper left corner of
  the document window.
```

```
Keyboard shortcut: AMIGA-% or %

See also:
              GETPART
              ,
              LOADLIB
              .

ARexx call: MENU("Choose component")    Warning: solid space beetween
          (ALT SPACE) the words Choose and component.
```

1.42 Drawing menu/Place component

```
               This entry enables the component mode. The current symbol is ↩
                  drawn
  under the cursor and you can place it on the document. If no symbol is
  valid, the
              bgui requester
               is opened.
Click with the left mouse button to place a component on the document.
  You are greatly encouraged to use the option
              Snap on grid
               to place the
  components.

Keyboard shortcut: AMIGA-H or H

See also:
              PUTPART
               ARexx call: MENU("Place component")     Warning: solid space ↩
                  beetween
          (ALT SPACE) the words Place and component.
```

1.43 Drawing menu/Place reference

```
               When this entry is marked, when a component will be
              placed
               with the
  mouse, his reference will also be placed next it.
This reference can be edited by a double click on the component or
  on the reference itself. You can also move this object at another place
  with the left mouse button (click on it, and drag the mouse while the
  button is switched).
Default references are defined in the libraries an can't be changed.

  You can call the
              ARexx script
               AddRefs to add the part numbers.

There is no keyboard shortcut for this entry, but you can also change
  its state under the bgui.requester.
```

See also:
SETREF
.

This menu can't be called by ARexx.

1.44 Drawing menu/Place value

When this entry is marked, when a component will be placed
with the
mouse, his default value will also be placed next it.
This value can be edited by a double click on the component or
on the value itself. You can also move this object at another place
with the left mouse button (click on it, and drag the mouse while the
button is switched). The default value is the name of the object.

There is no keyboard shortcut for this entry.

See also:
SETVAL
.

This menu can't be called by ARexx.

1.45 Drawing menu/Place pins numbers

When this entry is marked, when a component will be placed
with the
mouse, his pins will be numbered, if this component is defined with
pin numbers in the library (allways in this case for IC).
If a component is placed on the document and if you want to modify
this indication, double click on it with the left mouse button and
click on the button "Pins numeration" to change this option, if it
can be edited (resistances for example have no pin number).

There is no keyboard shortcut for this entry.

This menu can't be called by ARexx.

1.46 Drawing menu/Rotation

When this entry is selected the current symbol or the selected
objects are rotated by 90 degrees. If you press a SHIFT key while
selecting the menu the rotation will be anticlockwise.
This operation can also be performed on a clip (just select the
menu Edit/Paste, then this one).

Keyboard shortcut : AMIGA-R or R (clockwise)

AMIGA-SHIFT-R or SHIFT-R (anticlockwise)

See also:
ROTATE
.

ARexx call: MENU("Rotation")

1.47 Drawing menu/Reflection

When this entry is selected the current symbol or the selected objects are reflected. This operation is done by the vertical axis of the objects (or horizontal if they were rotated).
This operation can also be performed on a clip (just select the menu Edit/Paste, then this one).

Keyboard shortcut : AMIGA-/ or /

See also:
SYMMETRY
.

ARexx call: MENU("Reflection")

1.48 Drawing menu/Alternate symbol

Select this menu to toggle the symbol definition of a component, when it exists in the library. You can perform this operation on the current symbol being placed or the selected components.

This feature is usefull for amplifiers for inverting the + and - inputs. In some libraries the alternate symbol is an american symbol (try the resistance).

Keyboard shortcut : AMIGA-~ or ~

See also:
CONVERT
.

ARexx call: MENU("Alternate")

1.49 Drawing menu/Normal position

This entry invalidate the effects of the precedents
rotation
,
reflection and
Alternate~symbol
.

You can perform this operation on the current symbol being placed
or on the selected components.

Keyboard shortcut : AMIGA-N ou N.

ARexx call: MENU("Normal~position") Warning: solid space beetween
(ALT SPACE) the words Normal and position.

1.50 Drawing menu/Place line

This entry enables the wire mode (or line mode). When it's selected
the cursor is changed (in a cross). To place a line, just click on the
beginning, click on the left mouse button, drop it, place now the cursor
at the end point and click again with the left button. Click on the
right button to abort a placement.
To stop this mode, click on the right button or select another mode.

The current line width is indicated in the
title bar
.
It can be modified using one of this menus:
Double line
,
Bus
,
Dashed line
or
Any width
.

To modify a line, click on an extremity with the left button, and
maintaining the button pushed drag the mouse at another place.

You can also use the keyboard to adjust a line: select this line by
clicking on it, and use the cursor keys with the ALT or CTRL keys to
modify the beginning or the ending of the line. You can also use the
SHIFT key to move faster (grid size).

Keyboard shortcut : AMIGA-SPACE or SPACE

See also:
Orthogonal mode
,
Continuous drawing
,
Double line
,

Bus
et
Dashed line
.

DRAW

.

ARexx call: MENU("Place~line") Warning: solid space beetween (ALT SPACE) the words Place and line.

1.51 Drawing menu/Orthogonal mode

When this entry is marked, the lines are allways placed horizontally or vertically, or in a direction of 45\textdegree{}.
Select this menu to toggle its state.

Keyboard shortcut : AMIGA-| or |

ARexx call: MENU("Orthogonal~mode") Warning: solid space beetween (ALT-SPACE) the words Orthogonal and mode.

1.52 Drawing menu/Continuous drawing

When this entry is marked, when the line mode is active, a new line is placed at the extremity of the precedent.
Click on the right mouse button to interrupt the continuity of the lines.

Select this menu to toggle its state.

There is no keyboard shortcut for this entry.

This menu can't be called by ARexx.

1.53 Drawing menu/Double line

When this entry is marked, the lines are drawn with 2 pixels ↩ width.

To stop this mode, select this entry again or choose another mode:

Bus
,
Dashed line
or
Any width
.

Note that the current line width is written in the title bar
.

Keyboard shortcut : AMIGA-\ or \

```
See also:
              DRAWMODE(1)
              .

ARexx call: MENU("Double~line")         Warning: solid space beetween
          the words Double and line.
```

1.54 Drawing menu/Bus

```
                When this entry is marked, the lines are drawn with 7 pixels  ↩
                   width.

To stop this mode, select this entry again or choose another mode:

              Double line
              ,
              Dashed line
               or
              Any width
              .

Note that the current line width is written in the
              title bar
              .

Keyboard shortcut : AMIGA-. or .

See also:
              DRAWMODE(2)
              .

ARexx call: MENU("Bus")
```

1.55 Drawing menu/Any width

```
                When this entry is marked, the lines can drawn at any width. A
requester ask you for the width you want. Enter a value beetween 1
and 255. All the objects that will be placed after this choice,
will be drawn with this line width.
If you select this entry while it's marked, the line width becomes 1.

There is no keyboard shortcut for this entry.

See also:
              DRAWMODE(2)
              .

This menu can't be called by ARexx.
```

1.56 Drawing menu/Dashed line

```
                When this entry is marked, the lines are drawn like dashed lines ↩
                    .
To cancel this mode, select it again or choose another mode: Double
line, Bus or Any Width.

Keyboard shortcut : AMIGA-: or :

See also:
              DRAWMODE(0)
              .

ARexx call: MENU("Dashed~line")     Warning: solid space (ALT-SPACE)
            beetween the words Dashed and line.
```

1.57 Drawing menu/Place box

```
When you select this entry, the cursor comes in box mode (like
  in line mode). Click on a corner of the box you want to draw with
  the left button mouse, go to the opposite corner and click again.
  If you want to cancel the operation, click on the right button.

The boxes are drawn using the current width.

To modify an existing box, click in a corner, and with the
  left button mouse allways switched, place this corner whenever
  you want and release the button. To cancel the operation, click
  on the right mouse button.

To modify the line width of a box, double click on the box
  and give a new value in the requester.

Keyboard shortcut: AMIGA-B or B

ARexx call: MENU("Place~box")       Warning: solid space (ALT-SPACE)
            beetween the words Place and box.
```

1.58 Drawing menu/Place ellipse

```
                When you select this entry, an ellipse is drawn under the cursor ↩
                    .
  To place an ellipse on the document, move the mouse and when this
  ellipse is where you want it to be, click on the left button.
You can also modify his diameters using the cursor keys (UP and
  DOWN for the vertical axis, LEFT and RIGHT for the vertical axis). If
  you use also the SHIFT key, the modification will be faster.
You can also use the + and - keys to modify the two diameters
  in the same time.

To modify an ellipse that is on the document, you can use a double
  click on it or click on one of his axis, near his border, and drag the
```

```
  mouse. If you press the CTRL key while you are performing this operation
  the ellipse will allways be drawn as a circle.

To modify the position of an existing ellipse, click in his center
  and drag the mouse whenever you want on the document.

To cancel this mode, choose another placement mode (lines, components
  or connectors...) or click on the right mouse button.

You can use the menu Rotation for a selected ellipse (Reflection has
  no sense).

Keyboard shortcut : AMIGA-E or E

See also:
              ELLIPSE
              .

ARexx call: MENU("Place~ellipse")       Warning: solid space beetween
          (ALT-SPACE) Place and ellipse.
```

1.59 Drawing menu/Place arc

```
                When you select this entry, an arc is drawn under the cursor. To
  place this arc on the document, place the cursor whenever you want and
  click on the left mouse button.
To cancel this mode, use a click on the right mouse button.

You can modify his dimensions using the cursor keys with or without
  the ALT, CTRL and SHIFT keys. With the ALT key you can change the
  radius, with CTRL you can change the angles, the SHIFT key gives
  faster changes.

To edit an existing arc, you can use a double click or select it and
  use the ALT, CTRL and SHIFT keys in combination with the cursor keys,
  like before. If you none of these qualifier keys is used you can move
  the arc when you use the cursor keys.

You can also use the menus Rotation and Reflection to modify the
  selected arc.

Keyboard shortcut : AMIGA-§ or §

See also:
              ARC
              .

ARexx call: MENU("Place~arc")           Warning: solid space
        (ALT-SPACE) beetween Place and arc.
```

1.60 Drawing menu/Placer connexion

When you select this entry a junction is drawn under the cursor.
To place it, click on the left mouse button.

To cancel this mode, click on the right mouse button.

Keyboard shortcut : AMIGA-J or J

See also:
JUNCTION
.

ARexx call: MENU("Place~junction") Warning: solid space
(ALT-SPACE) beetween Place and junction.

1.61 Drawing menu/Place text

A requester is opened to let you give a text to place on the
document. If you valid this requester, the text is placed under
cursor and you can place where you want, just use the left mouse
button to fix it.
To cancel this mode, use the right mouse button.

You can rotate or reflect a text before placing it.

To modify a text, double click on it and choose a gadget in
the requester.

Keyboard shortcut : AMIGA-T or T

See also:
WRITE
.

ARexx call: MENU("Place text") Warning: solid space
(ALT-SPACE) beetween Place and text.

1.62 Drawing menu/Place input connector

A requester is opened to ask you for the connector name. This
object is attached to its right. The size of this object
depends on the length of its name. Use the left mouse button
to fix it on the document.
To cancel this mode, use the right mouse button.

To modify an existing object of this type double click on it
with the left mouse button, then choose a gadget in the requester.

If you want a right pointing arrow, choose the menu
Drawing/Reflection
.

```
Keyboard shortcut : AMIGA-< or <

See also:
              INPUT
              .

ARexx call: MENU("Place<~input~connector")      Warning: solid spaces
            (ALT-SPACE) beetween the words.
```

1.63 Drawing menu/Place output connector

```
                A requester is opened to ask you for the connector name. This
  object is attached to its left. The size of this object
  depends on the length of its name. Use the left mouse button
  to fix it on the document.
To cancel this mode, use the right mouse button.

To modify an existing object of this type double click on it
  with the left mouse button, then choose a gadget in the requester.

If you want a left pointing arrow, choose the menu
              Drawing/Reflection
              .

Keyboard shortcut : AMIGA-> or >

See also:
              OUTPUT
              .

ARexx call: MENU("Place~output~connector")      Warning: solid spaces
            (ALT-SPACE) beetween the words.
```

1.64 Drawing menu/Redraw all

```
  When you select this entry, all the document is cleared and redrawn.
  It's usefull after some selections, block moves, or to display the
  grid correctly after some editions...

Keyboard shortcut : AMIGA-Z or Z

ARexx call: MENU("Redraw")
```

1.65 Menu Édition

```
                Cette série de menus est relative aux blocs d'éléments ↩
                   sélectionnés.
  Avant de commencer leur description il faut expliquer comment sont
  sélectionnés ces blocs.
Cette sélection peut s'effectuer de deux manières :
```

- en cliquant dans une zone ne contenant aucun élément puis en maintenant le bouton gauche de la souris, et en faisant glisser celle-ci vous pouvez encadrer une zone dans laquelle tous les éléments présents seront sélectionnés,
- en cliquant sur un élément, celui-ci sera sélectionné. Cependant si le menu

Édition/Multisélection

n'est pas marqué, seul un élément peut être sélectionné, sauf si vous enfoncez une des touches SHIFT au moment où vous cliquez.

Quand un bloc est déjà sélectionné, vous pouvez l'étendre en pressant une des touches SHIFT avant de cliquer avec le bouton gauche de la souris, pour n'importe quelle des deux méthodes vues précédemment.

Pour annuler la sélection d'un élément ou d'un bloc, pressez la touche Ctrl en même temps que vous utilisez la souris (lors du glissement).

Pour annuler la sélection d'un bloc pressez la touche ESC ou cliquez à nouveau deux fois de suite n'importe où sur l'écran.

Copier

Coller

Couper

Effacer

Cloner

Fixer sur grille

Passer devant

Passer derrière

Doubler taille

Diviser taille

Grouper

Séparer

Sauver clip

Charger clip

Multisélection

Restaurer

1.66 Menu Édition/Copier

```
                  Ce menu permet la recopie du bloc spécifié dans une zone ↩
                     mémoire.
  Si l'opération réussit, la sélection est annulée.
  Le programme gère 10 zones mémoires différentes permettant ainsi
  de stocker 10 blocs différents (voir
              CLIPUNIT
              ).

Keyboard shortcut : AMIGA-C (copier) ou C

See also: fonctions
              COPY
              ,
              PASTE
              .

ARexx call: MENU("Copier")
```

1.67 Menu Édition/Coller

```
                  Le contenu de la zone mémoire valide est copié à l'emplacement ↩
                      actuel
  du curseur. Il se déplace avec la souris, pour placer ce bloc, cliquez
  sur le bouton gauche.

Keyboard shortcut: AMIGA-V ou V

See also : fonctions
              CLIPUNIT
              ,
              PASTE
              ,
              COPY
              .

ARexx call: MENU("Coller")
```

1.68 Menu Édition/Couper

```
                  Ce menu permet d'effacer le bloc spécifié (le bloc est copié  ↩
                     dans la
  zone mémoire alors valide).

  Warning: cette fonction peut modifier les numéros des objets restants.

Keyboard shortcut : AMIGA-X ou X

See also: fonctions
              CLIPUNIT
              ,
```

PASTE
,
COPY
.

ARexx call: MENU("Couper")

1.69 Menu Édition/Effacer

Ce menu permet d'effacer le bloc spécifié. Vous pouvez aussi utiliser la touche DEL.

Warning: cette fonction peut modifier les numéros des objets restants.

Keyboard shortcut : AMIGA-Y, Y ou DEL

ARexx call: MENU("Effacer")

1.70 Menu Édition/Cloner

Ce menu permet de recopier le bloc spécifié.

Keyboard shortcut : AMIGA-TAB ou TAB

ARexx call: MENU("Cloner")

1.71 Menu Édition/Fixer sur grille

Ce menu permet d'aligner les éléments appartenant au bloc spécifié sur la grille (leur point d'attache, déterminé par le point correspondant au pointeur de la souris, lors de leur pose, est fixé sur le point le plus proche de la grille).

Keyboard shortcut : AMIGA-! ou !

ARexx call: MENU("Fixer")

1.72 Menu Édition/Passer devant

Ce menu permet de faire passer un élément en tête de la liste, il est alors choisi s'il est en recoupement avec d'autres quand l'utilisateur clique sur eux.

À noter une astuce pour sélectionner un élément qui est en recoupement avec d'autres, alors qu'il est derrière: sélectionnez d'abord le premier élément en cliquant dessus puis utilisez la touche ALT lorsque vous cliquez avec le

bouton gauche, ce sera l'élément suivant qui sera sélectionné.

Warning: cette fonction modifie les numéros des objets.

Keyboard shortcut : AMIGA-M ou M

ARexx call: MENU("Passer~devant") Warning: espace "solide"
(ALT-ESPACE) entre Passer et devant.

1.73 Menu Édition/Passer derrière

Ce menu permet de faire passer un élément en queue de
liste. Il est alors choisi en dernier s'il est en recoupement
avec d'autres quand l'utilisateur clique sur eux.

Warning: cette fonction modifie les numéros des objets.

Keyboard shortcut : AMIGA-D ou D

ARexx call: MENU("Passer~derrière") Warning: espace "solide"
(ALT-ESPACE) entre Passer et derrière.

1.74 Menu Édition/Doubler taille

Ce menu permet de doubler la taille des éléments sélectionnés.
Cette fonction s'applique aussi à l'élément présent sous le
curseur lorsque celui-ci est en mode "placement composant".

La fonction ne prend effet que si l'agrandissement est possible,
c'est à dire sans que les éléments ne sortent de la fenêtre.

Keyboard shortcut : AMIGA-+ ou +

See also: fonction
SETSCALE
.

ARexx call: MENU("Doubler~taille") Warning: espace "solide"
(ALT-ESPACE) entre Doubler et taille.

1.75 Menu Édition/Diviser taille

Ce menu permet de diminuer la taille des éléments sélectionnés ↩
.
La taille des éléments doit être un multiple de deux (échelles 2,
4, 8...)
Cette fonction s'applique aussi à l'élément présent sous le
curseur lorsque celui-ci est en mode "placement composant".

```
Keyboard shortcut : AMIGA-- ou -

See also: fonction
              SETSCALE
              .

ARexx call: MENU("Diviser~taille")     Warning: espace "solide"
        (ALT-ESPACE) entre Diviser et taille.
```

1.76 Menu Édition/Grouper

```
  Ce menu permet d'associer les éléments sélectionnés, en un
  groupe, leur sélection se fait alors en cliquant sur un seul
  d'entre eux, cependant leur édition individuelle reste toujours
  possible, s'il y a lieu, en cliquant deux fois dessus.
  Il est impossible d'annuler la sélection d'un seul élément d'un
  groupe : la sélection d'un de ses éléments ou son annulation est
  transmise automatiquement aux autres éléments du groupe.

Keyboard shortcut : AMIGA-{ ou {

ARexx call: MENU("Créer~groupe")       Warning: espace "solide"
        (ALT-ESPACE) entre Créer et groupe.
```

1.77 Menu Édition/Séparer

```
  Ce menu permet de supprimer les liens définissant un groupe
  sélectionné, les éléments retrouvent alors leur indépendance.
  Les éléments sont désélectionnés pour marquer la réussite
  de la fonction. Ceux qui n'appartiennent pas à un groupe
  restent sélectionnés.

Keyboard shortcut : AMIGA-} ou }

ARexx call: MENU("Dissocier~groupe")   Warning: espace "solide"
        (ALT-ESPACE) entre Dissocier et groupe.
```

1.78 Menu Édition/Sauver clip

```
                  Ce menu permet de sauvegarder le bloc sélectionné dans un
  fichier (choisi à l'aide d'une requête).

Keyboard shortcut : AMIGA-W ou W

See also: fonctions
              SAVECLIP
              ,
              LOADCLIP
              .
```

ARexx call: MENU("Sauver~clip") Warning: espace "solide"
(ALT-ESPACE) entre Sauver et clip.

1.79 Menu Édition/Charger clip

Ce menu permet de lire un bloc précédemment sauvé dans un fichier (choisi à l'aide d'une requête). Si la lecture réussit le bloc de données est affiché sous le curseur et se déplace avec lui, utilisez un clic du bouton gauche pour le fixer à l'emplacement souhaité ou bien cliquez avec le bouton droit pour annuler l'opération.

Le point d'accrochage d'un clip est le point le plus haut à gauche, aussi essayez de créer des clips possédant ce point situé sur la grille afin de pouvoir les placer facilement par la suite.

Keyboard shortcut : AMIGA-L ou L

See also: fonctions
LOADCLIP
,
SAVECLIP
.

ARexx call: MENU("Charger~clip") Warning: espace "solide"
(ALT-ESPACE) entre Charger et clip.

1.80 Menu Édition/Multisélection

Quand ce menu est marqué, vous pouvez sélectionner autant d'éléments que vous le désirez, en cliquant successivement sur chacun d'eux, sans avoir à utiliser une touche SHIFT. Quand il n'est pas marqué, vous devez utiliser une touche SHIFT pour sélectionner plusieurs éléments, sinon chaque nouvelle sélection annule la précédente.

Il suffit de sélectionner ce menu pour changer son état.

Keyboard shortcut : AMIGA-*

Pas d'appel ARexx possible pour ce menu.

1.81 Menu Édition/Restaurer

Ce menu est très pratique pour annuler l'effet de la dernière opération qui a été réalisée. Cependant les effets réalisés à l'aide des fonctions ARexx ne pourront être annulés. D'autre part ce menu est quelque peu bugué...

```
Keyboard shortcut : AMIGA-U ou U

ARexx call: MENU("Restaurer")
```

1.82 Menu Macros

```
                Cette série de menus est dédiée aux
               macro-commandes
                et
  aux
               scripts ARexx
               .


               Mode direct
                      Saisie, exécution d'une ligne de commande.

               Appel~script
                      Appel d'un script ARexx choisi.

               ARexx...
                      Appel (programmable) de 10 scripts.
```

1.83 Menu Macros/Mode direct

```
                   Ce menu permet la saisie d'une
               macro-commande
               , puis son exécution. À noter
  que vous pouvez également utiliser la combinaison d'une touche ALT et d'une
  touche de fonction pour obtenir 10 macros préprogrammées (voir

               touches de fonction
               ).
  Il faut noter que ces séquences programmées sont sauvegardées
  dans le fichier "s:AmiCAD.keys", à l'aide du menu

               Preferences/Touches/Sauver
               . Celles-ci sont alors rechargées lors de
  chaque lancement du programme.

  Vous pouvez provoquer l'affichage du résultat de la commande en
  faisant commencer le texte spécifiant cette commande par le signe
  égal (=).
Exemple: =4*95      affiche le résultat de la multiplication (4 fois 95)
   =FIRSTSEL  affiche le numéro du premier objet sélectionné

Keyboard shortcut : AMIGA-; ou ;

See also: fonctions
               EXEC
               ,
               MACRO
```

,
MAP
.

ARexx call: MENU("Mode~direct") Warning: espace "solide"
(ALT-ESPACE) entre Mode et direct.

1.84 Menu Macros/Appel script

Ce menu permet la saisie d'une commande ARexx (donnez le nom ↩
du
script
,
l'extension ".amiCAD" n'est pas obligatoire).
Keyboard shortcut : AMIGA-, ou ,

See also: fonction
CALL
.

ARexx call: MENU("Appel~script") Warning: espace "solide"
(ALT-ESPACE) entre Appel et script.

1.85 Menu Macros/ARexx...

Ces 10 menus servent à définir 10 commandes pouvant être ↩
lancés par la
sélection du menu correspondant.
Lors du premier appel du menu, l'utilisateur doit définir sa commande,
en donnant le nom du script à exécuter(sans le chemin).
Les appels qui suivront permettent alors l'appel du
script
défini.
Le nom du script apparaît ensuite dans le menu. Les 10 définitions peuvent
être sauvegardées dans le fichier "AmiCAD.prefs", et rechargées
automatiquement lors de l'appel du programme.
Ces menus peuvent être redéfinis en appuyant sur l'une des touches
SHIFT lors de la sélection du menu.

À noter que l'édition du script défini peut être faite de façon
automatique dans une nouvelle fenêtre en appelant le menu tout en
maintenant la touche CTRL appuyée (le nom doit naturellement avoir
été défini auparavant). Le programme lance alors l'exécution du script
ARexx EditScript.AmiCAD en lui passant le nom de ce script comme
argument. Ce script (EditScript) doit être adapté à l'éditeur que
vous utilisez et à votre installation.

La commande doit comporter le nom du script à exécuter (toujours sans
l'extension ".amiCAD"). Le nom ne peut compter que 11 caractères au maximum.
Il n'est pas possible de transmettre de paramètres au script.
Les communications avec ARexx se font grâce au
port

```
              AMICAD
   (ou AMICAD0, AMICAD1...AMICAD9 si plusieurs tâches sont exécutées en
   même temps). Utilisez la commande port=ADDRESS() pour connaître le nom
   du port, si nécessaire.
Raccourcis clavier : AMIGA-0 à AMIGA-9

See also: fonction
              CALL
              .

Pas d'appel ARexx possible pour ces menus, utilisez la fonction CALL.
```

1.86 Port ARexx

Le programme ouvre normalement un port de communication avec l'interpréteur ARexx lors de son lancement, ce port s'appelle AMICAD. Si le programme est lancé plusieurs fois, le port est appelé AMICAD0 par la seconde tâche, puis AMICAD1 pour la suivante, et ainsi de suite jusqu'à AMICAD9.

1.87 Menu Preferences

```
                Cette série de menus permet de sauvegarder certains réglages ↩
                   du
programme dans les fichiers AmiCAD.prefs et AmiCAD.keys. Ces
fichiers sont sauvés dans le répertoire courant.
Ces fichiers sont relus automatiquement à chaque démarrage du
programme, afin de repositionner les réglages comme voulu par
l'utilisateur. Ces deux fichiers sont au format ASCII et peuvent
donc être consultés et même modifiés à l'aide d'un éditeur de
texte. Vous pouvez notamment créer des minis-fichiers de
configuration ne contenant qu'une partie de la configuration,
comme une palette par exemple (voir le fichier AmiCAD.prefs.colors
pour l'exemple).


            Défilement~auto

            Copie~schéma~sauvé

            Sauver~icône

            Tirer~lignes

            Afficher grille

            Échelle horizontale

            Échelle verticale

            Dimensions document

            Choix mode écran
```

Palette

Choix fonte

Configuration

Touches

1.88 Menu Preferences/Défilement auto

Quand ce menu est marqué, si le curseur arrive à une bordure
de la fenêtre de travail, le défilement du contenu de la
feuille de schéma se fait automatiquement s'il y a lieu.

There is no keyboard shortcut for this entry.

Pas d'appel ARexx possible pour ces menus.

1.89 Menu Preferences/Copie schéma sauvé

Quand ce menu est marqué, lors de la sauvegarde, l'ancienne version
est sauvegardée avec l'extension ".bis" à la fin du nom.

There is no keyboard shortcut for this entry.

See also: fonction
SAVECOPY
.

Pas d'appel ARexx possible pour ce menu.

1.90 Menu Preferences/Sauver icône

Si ce menu est marqué une icône est créée lors de la sauvegarde
du texte. Le nom du programme figure dans le champ TOOL-TYPES.
L'icône utilisée est l'icône "Projet" par défaut, stockée dans
le répertoire ENV:.

There is no keyboard shortcut for this entry.

See also: fonction
SAVEICON
.

Pas d'appel ARexx possible pour ce menu.

1.91 Menu Preferences/Tirer lignes

Si ce menu est marqué, lors du déplacement d'un ou plusieurs composants à l'aide la souris, les fils connectés à ces composants seront également déplacés. Warning : cette opération n'a lieu qu'après avoir lâché ces composants à leur emplacement de destination.

Keyboard shortcut: AMIGA-F ou F

ARexx call: MENU("Tirer")

1.92 Menu Preferences/Afficher grille

Si ce menu est marqué, une grille est affichée sur l'écran. Le pas utilisé est celui défini dans le menu
Dessin/Pas grille
.
L'écran est mis à jour après l'appel du menu.

There is no keyboard shortcut for this entry.

Pas d'appel ARexx possible pour ce menu.

1.93 Menu Preferences/Nom complet

Ce menu permet d'afficher le nom complet du document dans la
barre de titre
, s'il est marqué. S'il ne l'est pas, le nom est affiché sans le chemin, ceci permet d'obtenir les indications qui suivent, les coordonnées du curseur notamment, quand la fenêtre n'est pas suffisamment large ou quand le nom complet est très long (répertoires et sous répertoires multiples).

1.94 Menu Preferences/Horizontal scale

La valeur associée à ce menu permet de définir l'échelle par défaut des composants qui seront placés sur le schéma.

There is no keyboard shortcut for this entry.

See also: fonctions
HSCALE
,
SETSCALE
.

ARexx call: MENU("Échelle~horizontale") Warning: espace "solide"
(ALT-ESPACE) entre Échelle et horizontale.

1.95 Menu Preferences/Vertical scale

La valeur associée à ce menu permet de définir l'échelle par défaut des composants qui seront placés sur le schéma.

There is no keyboard shortcut for this entry.

See also : fonctions
VSCALE
,
SETSCALE
.

ARexx call: MENU("Échelle~verticale") Warning: espace "solide"
(ALT-ESPACE) entre Échelle et verticale.

1.96 Menu Preferences/Dimensions document

Ce menu permet de définir la largeur et la hauteur de la fenêtre de travail. La fenêtre est en fait du type SuperBitmap, ce qui permet de bénéficier de défilements très rapides, mais exige une quantité mémoire "chip" importante.

There is no keyboard shortcut for this entry.

See also: fonction
DIMSHEET
.

ARexx call: MENU("Dimensions")

1.97 Menu Preferences/Choix mode écran

Ce menu permet de choisir le type d'écran utilisé pour afficher les fenêtres. Une requête asl est affichée pour choisir ce mode. Les modes basse-résolution permettent de travailler sur certains détails alors que les modes haute-résolution permettent d'avoir une vue d'ensemble du schéma.

Raccourcis clavier : AMIGA-] ou]

See also: fonction
SCREEN
.

```
ARexx call: MENU("Choix~mode")          Warning: espace "solide"
        (ALT-ESPACE) entre Choix et mode.
```

1.98 Menu Preferences/Palette

```
  Ce menu permet de choisir les couleurs de l'écran utilisé par AmiCAD.
  À noter que ces couleurs sont sauvées dans le fichier de configuration.
  Cette fonction fait appel à la bibliothèque reqtools.library pour
  fonctionner. Assurer-vous que vous en avez une copie dans votre
  répertoire "libs:".

  Warning: n'utilisez pas une couleur trop sombre pour la couleur 0,
  du noir par exemple, car alors le curseur utilisé pour placer les fils
  deviendrait invisible...

There is no keyboard shortcut for this entry.

ARexx call: MENU("Palette")
```

1.99 Menu Preferences/Choix fonte

```
                  Ce menu permet de choisir la fonte utilisée dans l'écran ↩
                     AmiCAD.
  En l'absence de fichier de configuration (voir ci-dessous), c'est la fonte
  choisie pour l'écran public par défaut qui est utilisée.

Raccourcis clavier : AMIGA-[ ou [

See also: fonction
              SCREEN
              .

ARexx call: MENU("Choix~fonte")         Warning: espace "solide"
        (ALT-ESPACE) entre Choix et fonte.
```

1.100 Menu Preferences/Configuration

```
                Ce menu comporte trois rubriques, permettant de sauver ou de
  lire les noms des
             scripts ARexx
              choisis, ainsi que
  les divers modes (défilement, sauver avec ou sans icône, copie schéma...).

CHARGER : permet de relire le fichier de configuration de votre choix,
préalablement sauvé grâce à l'option SAUVER ou SAUVER EN.

SAUVER  : permet la sauvegarde des préférences dans le fichier
AmiCAD.prefs. Ce fichier est relu à chaque chargement du
programme. Il doit être situé dans le répertoire courant.
```

SAUVER EN : permet la sauvegarde des préférences dans un fichier, de votre choix (requête asl.library).

See also: fonctions
LOADPREF
,
SAVEPREF
.

Pas d'appel ARexx pour ces menus.

1.101 Menu Preferences/Touches

Ce menu comporte trois rubriques, permettant de sauver ou de lire les séquences de touches programmées (touches de fonction, combinaisons de touches).

CHARGER : permet de relire le fichier de configuration de votre choix, préalablement sauvé grâce à l'option SAUVER ou SAUVER EN.

SAUVER : permet la sauvegarde des préférences dans le fichier AmiCAD.keys. Ce fichier est relu à chaque chargement du programme.

SAUVER EN : permet la sauvegarde des préférences dans un fichier, de votre choix (requête asl.library).

See also: fonctions
LOADKEYS
,
SAVEKEYS
,
MAP
,
Mode direct
.

Pas d'appel ARexx pour ces menus.

1.102 Format of the configuration files

Les fichiers AmiCAD.prefs et AmiCAD.keys sont des fichiers au format ASCII, ils peuvent ainsi être consultés et (aussi modifiés) à l'aide d'un simple éditeur de texte.
Leur format répond cependant à quelques règles élémentaires.

Chacun de ces deux fichiers commence par un en-tête qui ne doit pas être changé, il comprend notamment le numéro de version de ces fichiers, AmiCAD ne reconnaît que le numéro de version en vigueur pour une version donnée: la version 1.2 de AmiCAD attend l'en-tête "AmiCADPrefs 1.1" pour le fichier AmiCAD.prefs et l'en-tête "AmiCADKeys 1.0" pour le fichier AmiCAD.keys.

Le fichier AmiCAD.prefs comprend ensuite plusieurs paragraphes,
ces paragraphes peuvent être supprimés pour créer une configuration
particulière, comprenant par exemple uniquement une palette de
couleurs (fichier AmiCAD.palette par exemple).
Chacun de ces paragraphes débute par un texte placé entre deux
crochets ([Menus_ARexx], [Screen], [Palette] et [Mode]), ces textes
sont suivis de plusieurs lignes comprenant chacune des informations
différentes. Tout ou partie de ces lignes peuvent être supprimées.

De la même façon le fichier AmiCAD.keys comprend deux paragraphes
[FunctionKeys] et [Macros]. Vous pouvez ajouter ou supprimer des
lignes dans chacun de ces paragraphes, à votre convenance.

Warning: les mots situés à gauche des signes = doivent être
écrits tels quels (pas de changement majuscules/minuscules).

1.103 Écriture d'une ligne de commande(s)

Les macro-commandes peuvent être exécutées en mode local ou bien ↩
par
l'intermédiaire d'un
script ARexx
. Vous pouvez vous reporter aux scripts
donnés en exemple, dans le répertoire ARexx. Ceux-ci doivent posséder
l'extension ".amiCAD" à leur nom.
Chacune de ces macro-commandes peut appeler l'une des fonctions
ARexx, ou même plusieurs. Certaines d'entre elles nécessitent un ou
plusieurs arguments, enfin la plupart d'entre elles retournent un résultat.
Le programme permet aussi de manipuler des
variables
(des types numérique
ou chaîne de caractères). De plus vous pouvez définir vos propres fonctions.
L'appel d'une fonction se fait en donnant son nom, suivi d'un ou
plusieurs arguments, entourés de parenthèses, éventuellement séparés par
des virgules.

Les opérateurs mathématiques classiques sont bien sûr disponibles :

opérateur	signe	priorité
élévation à une puissance :	^	10
division :	/	9
multiplication :	*	9
modulo :	%	9
addition :	+	8
soustraction :	-	8
ET logique :	&	6
OU exclusif :	?	5
OU logique :	|	4
affectation :	=	2

Ce dernier opérateur (=) peut être utilisé pour affecter les
variables
.
Ex : A=2, affecte la valeur 2 à la variable A.
Il est à noter que l'écriture A=B=3 n'est pas admise, elle entraînera

```
  un message d'erreur "Affectation impossible". Utilisez plutôt l'écriture
  A=3:B=3

  Autres opérateurs :
décalage à gauche : <<
décalage à droite : >>
Ces deux opérateurs ont une priorité égale à 7.
test supérieur : >      test supérieur ou égal : >=
test inférieur : <      test inférieur ou égal : <=
test si différent : <>      test égalité : ==
Les tests renvoient la valeur 1 s'ils sont vérifiés, 0 dans les
  autres cas. Leur priorité est égale à 3.

  Il existe un opérateur particulier qui en fait n'en est pas un, c'est le
  signe :, celui-ci permet en fait de séparer deux formules (ou davantage)
  afin de pouvoir saisir plusieurs affectations de
              variables
               dans une même
  fonction. Ainsi vous pouvez saisir A=2:B=3:C=0 dans une seule commande au
  lieu d'en utiliser trois. Le résultat renvoyé est alors celui de la
  dernière opération réalisée (0 dans cet exemple). Cette possibilité peut
  également présenter un intérêt pour une fonction FOR (voir plus loin),
  afin de réaliser de multiples initialisations au début d'une boucle.

  Enfin il est à noter que vous pouvez placer un commentaire dans une macro
  comportant une expression mathématique en utilisant l'apostrophe vue
  plus haut.
Exemple:
    DELETE(FIRSTSEL) ' Effacement du premier élément sélectionné

Enfin, vous pouvez placer plusieurs appels de fonctions sur une
  même ligne, en les séparant par deux points.
Exemple:
    HSCALE(FIRSTSEL,2) : VSCALE(FIRSTSEL,2)
```

1.104 Les variables

```
                Les variables manipulées par AmiCAD peuvent être de deux types:
numérique ou chaîne de caractères. Le type est choisi lors de
l'affectation, il ne peut ensuite être changé, à moins de
réinitialiser cette variable (voir fonction
              INIT
              ).

Pour affecter une valeur à une variable il suffit de faire suivre
son nom du signe = et de la valeur à lui affecter.

Exemples:
    A = 1
    B2 = "chaîne de texte"
    A2 = "Première "+B2     donne "Première chaîne de texte"

Les chaînes de caractères doivent être encadrées par des guillemets
s'il s'agit de constantes. Les
              valeurs numériques
```

sont limitées aux
entiers (longs), utilisez ARexx pour manipuler des nombres réels.

Les noms de variables peuvent comprendre de 1 à 21 caractères, ils doivent débuter par une lettre, les caractères suivants pouvant être des lettres (accentuées ou non), des nombres ou le caractère _.

Exemples de noms valides:
    ESSAI
    TYPE_DONNÉE
    LIGNE1

À noter que le programme convertit normalement les noms de fonctions et de variables en majuscules lors de la saisie, aussi n'y aura-t-il pas de différence entre deux noms comme variable et VARIABLE (ou même Variable), cependant dans un
script ARexx
, utilisez de préférence les
majuscules car les chaînes issues de l'interpréteur ne sont pas forcément converties, et le programme différenciera alors les noms écrits en minuscules et/ou en majuscules.

Warning: les variables ne sont pas utilisables directement avec ARexx, elles sont internes au programme AmiCAD et l'interpréteur ARexx possède ses propres variables. Il est cependant possible de renvoyer une valeur de variable à l'interpréteur simplement en donnant son nom:
    Exemple:
  'NOM_VARIABLE'; v=result
  Le contenu de la variable NOM_VARIABLE est ainsi transféré dans
  la variable ARexx v.
    Il est souvent avantageux d'utiliser des variables internes à AmiCAD car leur traitement est plus rapide, d'autre part cela peut éviter l'envoi de certains messages entre l'interpréteur et le programme, ce qui entraîne un gain de vitesse énorme (voir scripts Zoom et Unzoom).

1.105 Les variables numériques

Les nombres manipulés par AmiCAD sont limités aux nombres entiers. Ceux ci sont des entiers longs signés, sur 32 bits. La valeur maximale est doc de 2^31-1 (2147483647) et la valeur minimale -2^31 (-2147483648). Utilisez les capacités de l'interpréteur ARexx pour manipuler des nombres en virgule flottante.

1.106 Les chaînes de caractères

Les chaînes de caractères comprennent un nombre quelconque de caractères (théoriquement limité uniquement par la capacité mémoire de votre micro-ordinateur), en commençant par les chaînes nulles qui ne comprennent aucun caractère. Pour définir une chaîne de caractères, vous devez encadrer son contenu par deux guillemets. Si cette chaîne comprend elle-même un ou plusieurs guillemets, ceux-ci doivent être

doublés.

Exemples:
"Ceci est une chaîne de caractères"
""
"C'est un guillemet "" !"

1.107 Structure des scripts ARexx

Les scripts ARexx sont des fichiers ASCII, se conformant à un format permettant leur interprétation par ce logiciel. Ils peuvent inclure toutes les fonctions spécifiques à ARexx et à ses bibliothèques (reportez-vous à leurs documentations pour plus de précisions), ainsi que toutes les commandes supportées par le programme AmiCAD (une centaine, plus les fonctions éventuellement définies par vos soins).

Emplacement
Ces scripts doivent se situer soit dans le répertoire AmiCAD/ARexx. Il est conseillé de les nommer avec l'extension .AmiCAD, mais ce n'est pas obligatoire.

Format
Ces scripts doivent toujours commencer par une remarque incluse entre les caractères /* et */, comme en langage C.

Les commandes AmiCAD doivent impérativement figurer en majuscules. Ce sont les mêmes que celles qui sont utilisées en
mode local
, cependant leur analyse par ARexx nécessite quelques précautions, ainsi ces commandes doivent impérativement être encadrées par des apostrophes (') ou des guillemets, afin de les différencier des fonctions internes à ARexx.

Exemple
La structure suivante est recommandée pour ces scripts:

```
/* Ce fichier vous donne un exemple de structure possible
pour un fichier de macros appelé par AmiCAD, par le menu Macros */

options results     /* indispensable pour récupérer le résultat des macros */

signal on error     /* pour l'interception des erreurs */
signal on syntax

/* votre programme doit être situé dans cette zone */

exit

/* Traitement des erreurs, interruption du programme */
syntax:
erreur=RC
```

```
'MESSAGE("Erreur de syntaxe"+CHR(10)+"en ligne 'SIGL'"+CHR(10)+"'errortext(erreur) ↩
    '")'
exit

error:
'MESSAGE("Erreur en ligne 'SIGL'")'
exit
```

Cet exemple existe déjà prêt dans le répertoire ARexx, sous le nom squelette.AmiCAD.

1.108 Liste des fonctions ARexx

```
ABS
 computes the absolute value of a number

ARC
 places an arc

ASC
 returns an ASCII code

ASK
 ask for a string from the user

BLINK
 draws and clears an object

CALL
 calls an ARexx script

CHR
 renvoie le caractère possédant le code ASCII spécifié

CLIPPATH
 choix du tiroir contenant les clips

CLIPUNIT
 choix de la zone mémoire utilisée pour les opérations couper/ ↩
    coller

CLOSE
 ferme la fenêtre texte spécifiée

COL
 numéro de colonne où est situé un objet

CONVERT
 choix symbole alterné pour composant

COPY
 copie d'un ou plusieurs objets en mémoire

DATE
```

```
 returns the actual date

DEF
 defines a new function

DELETE
 suppression d'un objet

DIMSHEET
 dimensionne la fenêtre (SuperBitmap)

DRAW
 tracé d'une ligne

DRAWMODE
 sélection du type de tracé

EDIT
 appel de la requête d'édition

ELLIPSE
 tracé d'une ellipse ou d'un cercle

EXEC
 interprete a string of chars

FILENAME
 nom du fichier complet

FILEPART
 nom du fichier

FINDPART
 look for a component

FINDREF
 recherche d'un composant par sa référence

FINDVAL
 recherche d'un composant par sa valeur

FIRSTSEL
 renvoie le numéro du premier objet sélectionné

FONTNAME
 renvoie le nom de la fonte de caractères utilisée

FONTSIZE
 renvoie la taille de la fonte de caractères utilisée

FOR
 traitement d'une boucle

GETPART
 choix du composant courant

GETREF
```

```
                lecture de la référence d'un composant

               GETVAL
                lecture de la valeur ou du type d'un composant

               HEIGHT
                renvoie la largeur d'un objet

               HELP
                displays an AmigaGuide help

               HSCALE
                renvoie la valeur de l'échelle horizontale d'un objet

               IF
                test

               INIT
                initialisation de variables

               INPUT
                placement d'un connecteur d'entrée

               JUNCTION
                place a junction

               LEN
                longueur d'une chaîne de caractères

               LIBSPATH
                choix du tiroir contenant les bibliothèques de symboles

               LINE
                numéro de ligne où est situé un objet

               LINKREF
                affectation d'une référence à un composant

               LINKVAL
                affectation d'une valeur à un composant

               LOAD
                chargement d'un schéma

               LOADCLIP
                chargement d'un clip
LOADKEYS chargement d'un fichier de macros

               LOADLIB
                chargement d'une bibliothèque de symboles

               LOADPREF
                chargement d'un fichier de préférences

               LOCK
                verrouillage saisies utilisateur
```

```
MACRO
 calls a programmed key sequence

MAP
 keyboard programming

MARK
 marquage d'un ou plusieurs objets

MARKZONE
 marquage des éléments compris dans la zone spécifiée

MENU
 executes the function associated to a menu

MESSAGE
 affichage d'un message

MESURE
  renvoie une dimension de la fenêtre

MODIF
 test modification texte

MOVE
 déplacement d'un objet

NBSHEET
 nombre de schémas présents en mémoire

NEW
 ouverture d'une nouvelle fenêtre

NEXTSEL
 numéro du prochain élément sélectionné

OBJECTS
 returns the number of objects

OPEN
 ouverture de schéma(s)

OUTPUT
 placement d'un connecteur d'entrée

PARTNAME
 lecture du nom d'un composant

PASTE
 collage du contenu d'un tampon mémoire

PENWIDTH
 établit la largeur du trait

PICKOBJ
 choix d'un objet à l'aide de la souris
```

```
PRINT
 impression du schéma

PUTPART
 placement d'un composant

READCONV
 lecture du type de symbole

READDEF
 reads the definition of a programmed function

READDEV
 lecture du numéro de circuit d'un composant

READMAP
 reads the definition of a programmed key

READTEXT
 lecture d'un texte associé à un objet

REMLIB
 suppression d'une bibliothèque

REQFILE
 choix d'un fichier

REQSHEET
 choix d'un schéma

REQUEST
 affichage d'un message, choix OUI/NON

RESET
 réinitialisation de variables

ROTATE
 rotation d'un objet

SAVEIFF
 sauvegarde au format IFF

SAVE
 sauvegarde d'un schéma

SAVECLIP
 sauvegarde d'un clip

SAVECOPY
 test/choix sauvegarde copie fichier

SAVEICON
 test/choix sauvegarde avec création icône
SAVEKEYS sauvegarde des macros

SAVEPREF
 sauvegarde fichier préférences
```

```
SCREEN
 choix du mode écran

SCRMODE
 renvoie le mode écran

SECURITY
 détermination d'un nombre de boucles maximum avant débordement

SELECT
 choix d'une option parmi plusieurs

SELFILE
 sélection d'un schéma par son nom

SELSHEET
 sélection d'un schéma par son indice

SETCOLOR
 choix couleur écran

SETDEV
 choix du circuit ou d'une porte de composant

SETGRID
 choix du pas de la grille

SETPINS
 affichage numéros de broches

SETREF
 fixe la référence d'un objet

SETSCALE
 sets the horizontal and vertical scales

SETTEXT
 fixe le texte d'un objet

SETVAL
 fixe la valeur ou le type d'un objet

SGN
 test du signe d'un nombre

SHEIGHT
 renvoie la hauteur de l'écran

STOBACK
 passage de l'écran en arrière-plan

STOFRONT
 passage de l'écran an avant-plan

STR
 conversion d'un nombre en chaîne de caractères (décimal)
```

```
SWIDTH
 renvoie la largeur de l'écran

SYMMETRY
 symétrie d'un objet

TEST
 test si objet sélectionné

TIME
 returns the actual hour

TITLE
 choix du titre affiché dans la fenêtre

TYPE
 renvoie le type d'un objet

TXHEIGHT
 renvoie la hauteur occupée par un texte

TXWIDTH
 renvoie la largeur occupée par un texte

UNLINK
 suppression des liens d'un composant

UNLOCK
 annule le verrouillage des actions utilisateur

UNMAP
 deletes a programmed key sequence

UNMARK
 annulation du marquage d'un objet

VAL
 conversion d'une chaîne de caractères en nombre

VERSION
 returns the version number of AmiCAD

VSCALE
 renvoie la valeur de l'échelle verticale d'un objet

WHEIGHT
 renvoie la hauteur totale du plan de travail

WHILE
 boucle tant_que...

WIDTH
 renvoie la largeur d'un objet

WINDOW
 dimensionne la fenêtre schéma
```

WRITE
 placement d'un texte

WTOBACK
 renvoie la fenêtre active en arrière-plan

WTOFRONT
 renvoie la fenêtre active en avant-plan

WWIDTH
 renvoie la largeur totale du plan de travail

Index thématique

1.109 Classement thématique des fonctions ARexx

 Traitement des
chaînes de caractères

 Placement de nouveaux objets sur le schéma

 Édition, modification des objets existants

 Fonctions portant sur des blocs d'objets

 Fonctions mathématiques

 Fonctions interactives

 Gestion des fenêtres

 Gestion des préférences

 Fonctions diverses

1.110 Fonctions ARexx de traitement des chaînes de caractères

ASC
 returns an ASCII code

```
CHR
 renvoie le caractère possédant le code ASCII spécifié

LEN
 longueur d'une chaîne de caractères

STR
 conversion d'un nombre en chaîne de caractères (décimal)

VAL
 conversion d'une chaîne de caractères en nombre
```

1.111 ARexx fonctions to place objects

```
ARC
 places an arc

BOX
 places a box

DELETE
 deletes an object

DRAW
 places a line

DRAWMODE
 select the current line mode

ELLIPSE
 place an ellipse

GETPART
 choose the current component

INPUT
 place an input connector

JUNCTION
 place a junction

OUTPUT
 place an output connector

PUTPART
 place a component

WRITE
 place a texte
```

1.112 Fonctions ARexx permettant de gérer les blocs d'objets

```
CLIPUNIT
 choix de la zone mémoire utilisée pour les opérations couper/ ↩
    coller

COPY
 copie d'un ou plusieurs objets en mémoire

FIRSTSEL
 renvoie le numéro du premier objet sélectionné

LOADCLIP
 chargement d'un clip

MARK
 marquage d'un ou plusieurs objets

MARKZONE
 marquage des éléments compris dans la zone spécifiée

NEXTSEL
 numéro du prochain élément sélectionné

PASTE
 collage du contenu d'un tampon mémoire

SAVECLIP
 sauvegarde d'un clip

TEST
 test si objet sélectionné

UNMARK
 annulation du marquage d'un objet
```

1.113 Fonctions ARexx permettant de gérer les objets

```
BLINK
 draws and clears an object

CLIPPATH
 choix du tiroir contenant les clips

COL
 numéro de colonne où est situé un objet

CONVERT
 choix symbole alterné pour composant

DELETE
 suppression d'un objet

EDIT
```

```
 appel de la requête d'édition

FINDPART
 look for a component

FINDREF
 recherche d'un composant par sa référence

FINDVAL
 recherche d'un composant par sa valeur

GETREF
 lecture de la référence d'un composant

GETVAL
 lecture de la valeur ou du type d'un composant

HEIGHT
 largeur d'un objet

HSCALE
 renvoie la valeur de l'échelle horizontale d'un objet

LIBSPATH
 choix du tiroir contenant les bibliothèques de symboles

LINE
 numéro de ligne où est situé un objet

LINKREF
 affectation d'une référence à un composant

LINKVAL
 affectation d'une valeur à un composant

LOADLIB
 chargement d'une bibliothèque de symboles

MOVE
 déplacement d'un objet

OBJECTS
 returns the number of objects

PARTNAME
 lecture du nom d'un composant

PENWIDTH
 établit la largeur du trait

READCONV
 lecture du type de symbole

READDEV
 lecture du numéro de circuit d'un composant

READTEXT
```

```
 lecture d'un texte associé à un objet

REMLIB
 suppression d'une bibliothèque

ROTATE
 rotation d'un objet

SETDEV
 choix du circuit ou d'une porte de composant

SETPINS
 affichage numéros de broches

SETREF
 fixe la référence d'un objet

SETSCALE
 sets the horizontal and vertical scales

SETTEXT
 fixe le texte d'un objet

SETVAL
 fixe la valeur ou le type d'un objet

SYMMETRY
 symétrie d'un objet

TYPE
 renvoie le type d'un objet

TXHEIGHT
 renvoie la hauteur occupée par un texte

TXWIDTH
 renvoie la largeur occupée par un texte

UNLINK
 suppression des liens d'un composant

VSCALE
 renvoie la valeur de l'échelle verticale d'un objet

WIDTH
 renvoie la largeur d'un objet
```

1.114 Fonctions ARexx de gestion des préférences

```
FONTNAME
 nom de la fonte de caractères utilisée

FONTSIZE
 taille de la fonte de caractères utilisée
```

```
LOADKEYS chargement d'un fichier de macros

              LOADPREF
               chargement d'un fichier de préférences

              SAVECOPY
               test/choix sauvegarde copie fichier

              SAVEICON
               test/choix sauvegarde avec création icône
SAVEKEYS sauvegarde des macros

              SAVEPREF
               sauvegarde fichier préférences

              SCREEN
               choix du mode écran

              SCRMODE
               renvoie le mode écran

              SETCOLOR
               choix couleur écran

              SETGRID
               choix du pas de la grille

              SHEIGHT
               renvoie la hauteur de l'écran

              SWIDTH
               renvoie la largeur de l'écran
```

1.115 Fonctions ARexx de calcul

```
              ABS
               computes the absolute value of a number

              FOR
               traitement d'une boucle

              IF
               test

              INIT
               initialisation de variables

              RESET
               réinitialisation de variables

              SECURITY
               nombre de boucles maximum avant débordement

              SGN
```

```
 test du signe d'un nombre

STR
 conversion d'un nombre en chaîne de caractères (décimal)

VAL
 conversion d'une chaîne de caractères en nombre

WHILE
 boucle tant_que...
```

1.116 Fonctions ARexx permettant le dialogue avec l'utilisateur

```
ASK
 ask for a string from the user

LOCK
 verrouillage saisies utilisateur

MESSAGE
 affichage d'un message

PICKOBJ
 choix d'un objet à l'aide de la souris

REQFILE
 choix d'un fichier

REQUEST
 affichage d'un message, choix OUI/NON

SELECT
 choix d'une option parmi plusieurs

UNLOCK
 annule le verrouillage des actions utilisateur
```

1.117 Fonctions ARexx diverses

```
CALL
 calls an ARexx script

DATE
 returns the actual date

DEF
 defines a new function

EXEC
 interprete a string of chars
```

```
FINDPART
 look for a component

HELP
 displays an AmigaGuide help

MACRO
 calls a programmed key sequence

MAP
 keyboard programming

MENU
 executes the function associated to a menu

READDEF
 reads the definition of a programmed function

READMAP
 reads the definition of a programmed key

TIME
 returns the actual hour

UNMAP
 deletes a programmed key sequence

VERSION
 returns the version number of AmiCAD
```

1.118 Fonctions ARexx de gestion des fenêtres

```
CLOSE
 ferme la fenêtre texte spécifiée

DIMSHEET
 dimensionne la fenêtre (SuperBitmap)

FILENAME
 nom du fichier complet

FILEPART
 nom du fichier

FONTNAME
 nom de la fonte de caractères utilisée

FONTSIZE
 taille de la fonte de caractères utilisée

LOAD
 chargement d'un schéma
```

```
MESURE
  renvoie une dimension de la fenêtre

MODIF
 test modification texte

NBSHEET
 nombre de schémas présents en mémoire

NEW
 ouverture d'une nouvelle fenêtre

OPEN
 ouverture de schéma(s)

PRINT
 impression du schéma

REQSHEET
 choix d'un schéma

SAVEIFF
 sauvegarde au format IFF

SAVE
 sauvegarde d'un schéma

SELFILE
 sélection d'un schéma par son nom

SELSHEET
 sélection d'un schéma par son indice

SETCOLOR
 choix couleur écran

SETGRID
 choix du pas de la grille

STOBACK
 passage de l'écran en arrière-plan

STOFRONT
 passage de l'écran an avant-plan

TITLE
 choix du titre affiché dans la fenêtre

WHEIGHT
 renvoie la hauteur totale de la fenêtre

WINDOW
 dimensionne la fenêtre schéma

WTOBACK
 renvoie la fenêtre active en arrière-plan
```

```
WTOFRONT
 renvoie la fenêtre active en avant-plan

WWIDTH
 renvoie la largeur totale du plan de travail
```

1.119 ARexx function DEF

```
                Vous pouvez définir un nombre quelconque de fonctions utilisant
  des fonctions internes ainsi que n'importe quel opérateur.
  Une de ces fonctions peut même faire appel à une autre fonction
  précédemment définie. Les arguments peuvent être d'un type quelconque,
  il doivent simplement correspondre aux types attendus par les fonctions
  qui seront appelées ou être compatibles avec les opérateurs utilisés.

Il ne peut y avoir plus d'une déclaration dans une même ligne.
La forme de ces définitions est de la forme suivante:
    DEF nom_fonction(argument,...) = définition des opérations.
Le mot clé DEF doit impérativement commencer la définition, sans
  aucun espace préalable.

Le nom des fonctions peut comprendre de 1 à 13 caractères
  alphanumériques, y compris les lettres accentuées et le soulignement (_).
  Le premier caractère doit cependant toujours être une lettre.
Les fonctions internes ne peuvent être redéfinies.
  Le nombre des arguments est limité à 15 au maximum.
Ce nombre est fixe pour une définition donnée (vous ne pouvez pas
  déclarer de fonction possédant un nombre variable d'arguments).
Il doit y avoir au moins un argument dans la définition, mais celui-ci
  peut ne pas être utilisé.
Les arguments doivent bien entendu posséder des noms distincts.

  Exemples:
DEF EXPAND(OBJET) = SETSCALE(OBJET,HSCALE(OBJET)+1,VSCALE(OBJET)+1)
L'appel de la fonction sera ensuite fait comme pour toute autre fonction:
    EXPAND(FIRSTSEL) par exemple.

La redéfinition d'une fonction ayant déjà été définie est possible,
  la nouvelle définition remplace alors l'ancienne. Ceci peut être utile
  lorsque vous essayez de définir une fonction complexe.

See also :

              READDEF
```

1.120 ARexx function ABS

```
                  ABS(number)
This function returns the absolute value of the specified number.
This number can be a
              variable
              .
```

```
Examples:
    ABS(-8)     returns 8
    ABS(4)      returns 4
    ABS(N)      returns the absolute value of the number in the N variable

Note: numbers can only be long integers.
```

1.121 ARexx function

```
                  ARC (x, y, horizontal_radius, vertical_radius, angle_begin,  ↩
                     angle_end)
Draws an arc. The center is specified by the two first arguments
x et y.

The angle_begin must be lower than the angle_end value. Their
units are degrees (negative values allowed).

If the operation is OK the function returns the number of the
new object.

Examples:
    ARC(100,100,25,25,0,90):ARC(100,100,25,25,90,180)   draws half a circle
    can be done with only one instruction:
    ARC(100,100,25,25,0,180)
    or
    ARC(100,100,25,25,-180,0)       (complementary)

    To draw a rounded box (x0,y0,x1,y1):  (all this lines have to placed on the  ↩
       same line)
    DEF ROUNDED_BOX(x0,y0,x1,y1)=
    ARC(x0+10,y0+10,10,10,-90,0):DRAW(x0+10,y0,x1-10,y0):ARC(x1-10,y0 ↩
       +10,10,10,0,90):
  DRAW(x1,y0+10,x1,y1-10):ARC(x1-10,y1-10,10,10,90,180):DRAW(x1-10,y1,x0+10,y1):
      ARC(x0+10,y1-10,10,10,180,270):DRAW(x0,y1-10,x0,y0+10)

See also: menu
              Drawing/Place arc
              ,
              ELLIPSE
```

1.122 ARexx function ASC

```
                  ASC("text",position)
This function returns the ASCII code of the character in the
text, the position of this character is given by the second
argument, this argument must be set beetween 1 (first char)
and the length of the text.

Examples:
    ASC("element",1)        returns 101 (ASCII code of char e)
    ASC(TEXT,LEN(TEXT))     returns the code of the last char
```

```
           in the
               variable
                TEXT.

See also:
               CHR
```

1.123 ARexx function ASK

```
  ASK("text")
This function opens a requester, the specified text is used for its
title. The user can enter a text in a cell or click on CANCEL.
The text is returned if the user clicks on OK, an empty string is
returned if the user clicks on CANCEL.
The specified text can have a few lines, separated by a line feed.

Example:
    ASK("Enter the first"+CHR(10)+"word then press"+CHR(10)+"the ENTER key")
```

1.124 ARexx function BLINK

```
                   BLINK(object_number)
The specified object is cleared and drawn three times.
The object number must be set beetween the values 1
and
               OBJECTS
               .
```

1.125 ARexx function BOX

```
                   BOX(x1, y1, x2, y2)
Place a box on the document, using the specified coordinates for
each of the opposite corners. The width of the lines depends on
the current line width (see
               DRAWMODE
               ).

If the placement is OK, the function returns the number of the new
object, else 0.

See also:
               DRAW
               .
```

1.126 ARexx function CALL

```
                  CALL("script name", argument1, argument2...)
Calls an ARexx script. The name of the script can be specified without
any path and extension ".amiCAD", if the script is in the ARexx drawer
of AmiCAD. This function can be used to call a script with some arguments
in a
              macro-command
               (ALT-Fx or direct mode).

You can use 0 to 15 arguments (
or
              strings of chars
              ).

The execution of the script is asynchronous.
The returned value is the script name...

Examples:
    'CALL("EditScript","Zoom")'
    'CALL("multiply",1.5,X)'
```

1.127 ARexx function CHR

```
                  CHR(code)
Renvoie le caractère possédant le code ASCII spécifié.

Utilisé pour obtenir un saut de ligne (CHR(10)) ou un caractère
spécial non présent sur le clavier.  Le code peut varier de 1 à
255 maxi.

Cette fonction peut aussi être utilisée pour écrire des caractères
spéciaux, qui ne font pas partie du jeu de caractères de l'Amiga.
Ainsi les caractères suivants sont utilisables:
    - 128: symbole de l'amplification (triangle pointé à droite)
    - 129: symbole collecteur ouvert
    - 130: symbole du OU (supérieur ou égal)
    - 131: flèche dirigée à droite
    - 133: symbole utilisé pour les entrées actives sur un front (supérieur)
    - 134: symbole de l'hystérésis
    - 135: symbole "trois états" (haute impédance)
    - 136: symbole d'une impulsion
    - 137: flèche dirigée à gauche
    - 138: symbole décrivant un signal analogique
    - 139: lettre correspondant au signe "ohm" (pour la valeur des résistances)
    - 140: lettre "alpha" (pour les potentiomètres)
    - 141: lettre TAU (constantes de temps)

Essayez ces codes à l'aide d'une macro du type
    WRITE(CHR(140),100,100)
Vous pouvez ajouter un de ces caractères (ou n'importe quel
autre à une chaîne de caractères avec une écriture de la
forme suivante:
    "Flèche à droite: "+CHR(131)

See also:
```

ASC

1.128 ARexx function CLIPPATH

CLIPPATH("chemin")
Détermine l'emplacement (le tiroir) où sont situés les clips.
Cette fonction permet de modifier le chemin acquis à l'aide de
l'outil
CLIPS
.

La fonction renvoie le chemin utilisé avant son utilisation.
Si le chemin spécifié est une chaîne nulle, il n'y a pas de
modification du chemin utilisé: CLIPPATH(""), seul le
chemin courant est renvoyé.

Exemple:
CLIPPATH("Work:AmiCAD/Clips")

1.129 ARexx function CLIPUNIT

CLIPUNIT(unité)
Cette fonction permet de choisir la zone mémoire
utilisée pour les opérations de couper/coller.
Ce numéro peut varier de 1 à 10.

La valeur renvoyée correspond au numéro de l'unité qui été
utilisée AVANT l'appel à cette fonction. Si vous voulez
connaître celui-ci sans en changer passez une valeur négative
ou nulle en argument.

Exemple de script:
'CLIPUNIT(1)' choix d'une nouvelle unité (numéro 1)
clip=RESULT conserver le numéro de l'ancienne unité
...
'CLIPUNIT('clip')' remettre l'ancienne unité

See also:
COPY
,
PASTE
.

1.130 ARexx function CLOSE

CLOSE(fenêtre)
Cette fonction provoque la fermeture de la fenêtre d'indice
spécifié. Chaque fenêtre possède un indice différent, en
commençant par l'indice 0 (voir
SELSHEET

```
              ).
La valeur renvoyée correspond au nombre de fenêtres texte
restant présentes en mémoire.

Utilisez la commande
              MENU
              ("Quitter") pour fermer toutes les
fenêtres et entraîner la fin du programme.

Pour cacher une fenêtre, sans perdre son texte en mémoire,
utilisez la commande MENU("Cacher"), pour la réduire à sa
taille minimale, utilisez la commande MENU("Réduire").

See also:
              OPEN
              ,
              LOAD
              .
```

1.131 ARexx function COL

```
                  COL(object_number)
Cette fonction permet de connaître le numéro de colonne où est
situé l'objet spécifié.
The object number must be set beetween the values 1
and
              OBJECTS
              .

See also:
              LINE
```

1.132 ARexx function CONVERT

```
                  CONVERT(object_number, 0/1/-1)
The object number must be set beetween the values 1
and
              OBJECTS
              .
S'il est nul, l'action validera le mode de placement courant
des composants (comme en utilisant le menu Dessin/
              Placer composant
                par exemple). La valeur renvoyée sera alors égale à 0 si le
mode de placement courant est le mode "normal", si le mode
"alterné" était valide, la valeur renvoyée est différente de 0.

Cette fonction permet de choisir le type de symbole pour un
composant. Selon la valeur du second argument, l'action sera
la suivante :
    - égal à 0: c'est le symbole normal qui sera utilisé,
    - égal à 1: c'est le deuxième symbole qui sera utilisé,
    - égal à -1: on change de symbole.
```

```
La valeur renvoyée est alors égale à 1 si l'objet sélectionné
est bien un composant, sinon c'est un 0.

See also: menu Dessin/
              Alterner~symbole
```

1.133 ARexx function COORDS

```
                  COORDS(object_number)
Cette fonction retourne les coordonnées de l'objet spécifié sous forme
de chaîne de caractères, séparées par des virgules.
Si l'objet est une ligne les coordonnées sont de la forme x0,y0,x1,y1
alors qu'elles sont de la forme x,y pour tout autre objet.

The object number must be set beetween the values 1
and
              OBJECTS
              .

Exemple d'utilisation:
    'COORDS('line')'; coord=result
    /* find the different coordinates */
    PARSE VAR coord x0 ',' y0 ',' x1 ',' y1

See also:
              COL
              ,
              LINE
              .
```

1.134 ARexx function COPY

```
                  COPY(clip)
Cette fonction permet de recopier les objets sélectionnés
dans la zone mémoire spécifiée. Le numéro de clip
peut varier de 1 à 10.
Utilisez la fonction
              PASTE
               pour coller les objets ainsi collés
ou choisissez l'unité comme unité par défaut (
              CLIPUNIT
              ) puis
utilisez le menu
              Édition/Coller
              .

La valeur renvoyée est égale à 1 si tout se passe normalement.
```

1.135 ARexx function DATE

```
                  DATE(jour)
Renvoie la date courante. Si l'argument jour est différent
de zéro, le jour de la semaine est inclus.

Exemples:
    DATE(0)     renvoie 18-Oct-97
    DATE(1)     renvoie Samedi 18-Oct-97

See also:
              TIME
              .
```

1.136 ARexx function DELETE

```
                  DELETE(object_number)
Supprime l'objet dont le numéro est spécifié.
Renvoie le nombre d'objets restants.

The object number must be set beetween the values 1
and
              OBJECTS
              .
```

Warning: cette fonction peut modifier les numéros des objets restants.

1.137 ARexx function DIMSHEET

```
                  DIMSHEET(largeur,hauteur)
Cette fonction permet de changer les dimensions de la
fenêtre de travail courante. Ces dimensions sont données
en pixels. Utilisez le menu
              Projet/Informations
               pour
connaître les dimensions de la feuille de travail.

Rappel: les fenêtres sont du type SuperBitmap, ce qui fait
que leur surface apparente peut être plus petite que leur
surface utile (il suffit de regarder l'état des ascenseurs
vertical et horizontal pour savoir si celle-ci est partiellement
cachée ou non). Ce type de fenêtre permet un défilement très
rapide du dessin lorsque le dessin est plus grand que l'écran,
cependant il peut être gourmand en mémoire chip, si vous
spécifiez de grandes dimensions. Les dimensions maximales
pouvant être prises par la fenêtre dépendent bien sûr de la
quantité de mémoire chip installée sur votre système.
Des dimensions de 1100 par 700 sont convenables pour un
système équipé de 2 Mo de chip.

À noter: la largeur est toujours ramenée à une valeur qui est
un multiple de 16, afin d'assurer une sauvegarde correcte du
dessin en utilisant le menu
```

```
              Projet/Sauver format IFF
              .

Valeur renvoyée: 1 si l'opération a réussi, 0 dans le cas contraire.
Warning, aucun contrôle n'est fait sur les valeurs passées en
arguments.

See also: menu
              Preferences/Dimensions document
              , fonctions
              WWIDTH
              ,
              WHEIGHT
```

1.138 ARexx function DRAW

```
                  DRAW(x0, y0, x1, y1)
Draw a line using the specified coordinates. The line
width is dependent of the valid mode (see
              DRAWMODE
              ).

If the line can be drawn, the function returns the object
number that have been placed, if there was an error 0 is
returned.
```

1.139 ARexx function DRAWMODE

```
                  DRAWMODE(type_ligne)
Détermine quel sera le type de ligne qui sera tracé (voir
              DRAW
              ).
Si type_ligne est égal à 0: ce seront des lignes en pointillés,
      égal à 1: ce seront des lignes "normales" (liaisons),
      égal à 2: ce seront des lignes doubles,
      égal à 3: ce seront des bus.
Si type_ligne est inférieur à 0 et supérieur à -256, ce seront des
lignes de largeur personnalisée qui seront tracées, avec une largeur
égale à la valeur absolue de celle qui a été spécifiée (v1.1).

Le menu Dessin est mis à jour en fonction du type sélectionné.

Cette fonction renvoie le type de ligne qui était en vigueur AVANT
l'appel de la fonction.
```

1.140 ARexx function EDIT

```
  EDIT(object_number)
Cette fonction appelle la requête permettant d'éditer un élément
du schéma. Elle a le même effet qu'un double clic sur un objet à
```

```
l'aide du bouton gauche de la souris.
Aucune valeur n'est renvoyée.

Cette fonction est utile  associée à une combinaison de  touches
du clavier.
Exemple:
    MAP("alt-e","EDIT(FIRSTSEL)")
    Il suffit alors  de sélectionner un objet puis  de faire  la
    combinaison  de touches  ALT-e  au  clavier  pour appeler la
    requête d'édition du premier élément sélectionné.
```

1.141 ARexx function ELLIPSE

```
                  ELLIPSE(x, y, horizontal_radius, vertical_radius)
Place an ellipse, the center is given by the two first args,
x et y, the radius by the others. If the placement is OK this
function returns the number of the new object, else 0.

The width of the line depends on the current line
width (see
              DRAWMODE
              ).

To draw a circle you can use the macro:
    DEF CIRCLE(x,y,r)=ELLIPSE(x,y,r,r)

See also:
              ARC
```

1.142 ARexx function EXEC

```
  EXEC("chaîne de caractères")
Demande l'interprétation et l'exécution de la chaîne de
caractères passée en argument, comme s'il s'agissait
d'une ligne de commande.
Le résultat dépend naturellement du contenu de la
chaîne passée en argument.

Exemple:
    EXEC(READTEXT(OBJET))   interprète la ligne de texte associée à l'objet ↩
       spécifié
    EXEC(READMAP("CTRL-)")  exécute la séquence associée à la combinaison de ↩
       touches CTRL-)
```

1.143 ARexx function FILENAME

```
                  FILENAME(name)
The current window is renamed with the file name given in
the argument. If it's an empty string ("") no renaming is
performed, it only returns the current name of the window
```

```
with complete path.

See also :
              FILEPART
```

1.144 ARexx function FILEPART

```
                  FILEPART("nom")
Renomme la fenêtre courante, en conservant le chemin.
Si une chaîne nulle ("") est passée en argument, cette
fonction renvoie le nom de la fenêtre courante, sans le
chemin complet.

Exemple:
    Supposons que le fichier courant soit "RAM:sources/Schéma"
    la fonction FILEPART("") renverrait Schéma, alors que la
    fonction FILEPART("Nouveau schéma") renommerait le fichier
    "RAM:sources/Nouveau schéma".

See also :
              FILENAME
```

1.145 ARexx function FINDPART

```
                  FINDPART(premier_objet, "composant")
Lance la recherche du composant dont le nom est spécifié sur
le document courant.
Ce nom correspond au nom du symbole, "RÉSISTANCE" par exemple
pour une résistance.
Le premier argument sert à commencer la recherche après
un objet déjà trouvé. Cet argument doit prendre une valeur
comprise entre 1 (la recherche s'effectuera sur tous les
objets du schéma) et
              OBJECTS
              .

La recherche s'effectue sans tenir compte de la casse des lettres,
c'est à dire que les majuscules et les minuscules ne sont pas
différenciées. D'autre part, vous pouvez inclure des "jokers" (ou
caractères génériques) dans la chaîne recherchée (voir documentation
AmigaDOS pour les diverses possibilités).

La valeur renvoyée est égale au numéro de l'objet qui a
été trouvé, ou à 0 si aucun objet ne correspond.

Exemples:
    'FINDPART(1,"RÉSISTANCE")'; objet=result
    'FINDPART(1,"(Rés|Cond)#?")'
```

1.146 ARexx function FINDREF

```
                  FINDREF(premier_objet, "référence")
Lance la recherche du composant dont la référence est
spécifiée, R3 par exemple pour une résistance ou CI9
pour un circuit intégré.
Le premier argument sert à commencer la recherche après
un objet déjà trouvé. Cet argument doit prendre une valeur
comprise entre 1 (la recherche s'effectuera sur tous les
objets du schéma) et
              OBJECTS
              .

La recherche s'effectue sans tenir compte de la casse des lettres,
c'est à dire que les majuscules et les minuscules ne sont pas
différenciées. D'autre part, vous pouvez inclure des "jokers" (ou
caractères génériques) dans la chaîne recherchée (voir documentation
AmigaDOS pour les diverses possibilités).

La valeur renvoyée est égale au numéro de l'objet qui a
été trouvé, ou à 0 si aucun objet ne correspond.

Exemples:
    'FINDREF(1,"R4")'; objet=result
    'FINDREF(1,"R#?")'; objet=result
    IF(O=FINDREF(1,"C2"),BLINK(O),0)

See also:
              FINDVAL
```

1.147 ARexx function FINDVAL

```
                  FINDVAL(premier_objet, "valeur")
Lance la recherche du composant dont la valeur ou le type
est spécifié, 10k par exemple pour une résistance ou 7400
pour un circuit intégré.
Le premier argument sert à commencer la recherche après
un objet déjà trouvé. Cet argument doit prendre une valeur
comprise entre 1 (la recherche s'effectuera sur tous les
objets du schéma) et
              OBJECTS
              .

La recherche s'effectue sans tenir compte de la casse des lettres,
c'est à dire que les majuscules et les minuscules ne sont pas
différenciées. D'autre part, vous pouvez inclure des "jokers" (ou
caractères génériques) dans la chaîne recherchée (voir documentation
AmigaDOS pour les diverses possibilités).

La valeur renvoyée est égale au numéro de l'objet qui a
été trouvé, ou à 0 si aucun objet ne correspond.

Exemples:
    'FINDVAL(1,"10k")'; objet=result
    IF(O=FINDVAL(1,"10$\mathrm{\mu}$F"),MARK(O),0)
```

```
See also:
              FINDREF
```

1.148 ARexx function FIRSTSEL

```
                  FIRSTSEL
Cette fonction renvoie le numéro du premier objet sélectionné.
Elle ne nécessite aucun argument. Si aucun objet n'est marqué
la valeur renvoyée est nulle.

See also:
              NEXTSEL
```

1.149 ARexx function FONTNAME

```
                  FONTNAME(x)
Renvoie le nom de la fonte utilisée dans la fenêtre courante.
L'argument peut prendre n'importe quelle valeur. Utilisez de
préférence l'argument
              SELSHEET
              (-1), en vue d'une compatibilité
ultérieure.

Exemple:
    FONTNAME(SELSHEET(-1))   renvoie topaz.font (par exemple)

See also:
              FONTSIZE
              .
```

1.150 ARexx function FONTSIZE

```
                  FONTSIZE(x)
Renvoie la taille de la fonte utilisée dans la fenêtre courante.
L'argument peut prendre n'importe quelle valeur. Utilisez de
préférence l'argument
              SELSHEET
              (-1), en vue d'une compatibilité
ultérieure.

Exemple:
    FONTSIZE(SELSHEET(-1))   renvoie 11 (par exemple)

See also:
              FONTNAME
              .
```

1.151 ARexx function FOR

```
                  FOR(init,condition_fin,action1,...) cette fonction permet de ↩
                     définir des
boucles. Le premier argument (init) est exécuté une seule fois,
quand l'appel de la fonction vient d'être fait. Le second argument
définit la condition de fin de boucle. Enfin le troisième argument
ainsi que les arguments suivants s'ils existent sont évalués à
chaque exécution de la boucle.

L'utilisation de cette fonction permet souvent un gain de temps
important dans l'exécution d'un script, le nombre de messages
pouvant diminuer de façon significative si elle est utilisée à
bon escient.

Exemples:
    FOR (I=0, I<10, I=I+1)
    Dans cet exemple la
              variable
               I est initialisée à la valeur 0,
    tant que cette valeur est inférieure à 10, on incrémente cette
    valeur. I va donc prendre successivement les valeurs 1 à 10.

    FOR (I=1, I<=10, I=I+1, DELETE(I))
    Cette formule comprend une instruction supplémentaire
    permettant de supprimer les 10 premiers objets.

    N=100:FOR (I=0:J=0, I<=N, J=J+I, I=I+1):J
    Cette formule permet de calculer la somme des 100 premiers
    nombres. Le résultat de la somme est renvoyé (:J à la fin).

Remarque: une boucle bloquée peut être interrompue par deux
moyens: soit par le nombre de boucles maximal (défini par la
fonction
              SECURITY
              , soit en appuyant simultanément sur les trois
touches CTRL, ALT et ESC.

See also:
              WHILE
              .
```

1.152 ARexx function GETPART

```
                  GETPART("nom_composant")
Permet de choisir le composant qui sera dessiné à l'aide
du menu "Dessin/Placer composant". Le nom du composant peut
figurer en majuscules comme en minuscules.
La fonction renvoie 1 si le composant a été trouvé, 0 sinon.

Exemples: GETPART("RÉSISTANCE")
    IF(GETPART(ASK("Composant?")),MENU("Placer~composant"),0)

À noter,  une information  intéressante:  le nom du composant
```

peut être spécifié par ses premières lettres seulement, c'est
en effet le premier composant dont le début du nom correspond
à l'argument qui sera alors choisi. Ainsi dans le second exemple
il suffit de saisir la lettre R dans la requête pour choisir une
résistance comme composant courant (si la bibliothèque de
symboles Symboles_AmiCAD est la première de la liste des
bibliothèques de symboles chargées).

See also:
LOADLIB
, menu
Dessin/Placer~composant

1.153 ARexx function GETREF

GETREF(object_number)
Renvoie le numéro de l'objet associé à la référence du composant
spécifié.
Le numéro de l'objet passé en argument doit naturellement être
un composant.
Si l'objet est d'un autre type la fonction renvoie -1.
Si cette référence n'existe pas la fonction renvoie 0.

The object number must be set beetween the values 1
and
OBJECTS
.

Exemple : READTEXT(GETREF(FIRSTSEL))

See also :
SETREF
,
LINKREF
,
UNLINK
,
GETVAL

1.154 ARexx function GETVAL

GETVAL(object_number)
Renvoie le numéro de l'objet associé à la valeur du composant
spécifié.
Le numéro de l'objet passé en argument doit naturellement être
un composant.
Si l'objet est d'un autre type la fonction renvoie -1.
Si cette référence n'existe pas la fonction renvoie 0.

The object number must be set beetween the values 1
and
OBJECTS

.

Exemple : READTEXT(GETVAL(FIRSTSEL))

See also :
SETVAL
,
LINKVAL
,
UNLINK
,
GETREF

1.155 ARexx function HEIGHT

HEIGHT (object_number)
Renvoie la hauteur de l'objet spécifié, en pixels.
The object number must be set beetween the values 1
and
OBJECTS
.

See also:
WIDTH

1.156 ARexx function HELP

HELP("node")
Cette fonction permet l'appel d'AmigaGuide, comme par le
menu
Projet/Aide
. L'argument doit être le
nom d'un noeud (node) du fichier AmiCAD.guide. Vous
pouvez, en particulier, spécifier n'importe quel nom
de fonction ou de menu.

Exemples:
HELP("Copy~to~clip")
HELP("DRAW")

1.157 ARexx function HSCALE

HSCALE(object_number)
Cette fonction renvoie la valeur de l'échelle horizontale de
l'objet spécifié.
The object number must be set beetween the values 1
and
OBJECTS
.

```
See also:
              VSCALE
              ,
              SETSCALE
              .
```

1.158 ARexx function IF

```
  IF(x, a1, a2)
Si x est différent de zéro, renvoie a1 sinon renvoie a2.
Seul a1 OU a2 sera évalué, selon le résultat de x. À noter que les
arguments a1 et a2 peuvent être de n'importe quel type, ils
peuvent également faire appel à d'autres fonctions, y compris
d'autres fonctions IF.

Exemples:
    IF(A>B,A,B)                 renvoie la valeur maximale de A et B
    DEF MIN(A,B)=IF(A<B,A,B)    définition fonction MIN
    IF(GETPART(ASK("Composant?")),MENU("Placer~composant"),0)
```

1.159 ARexx function INIT

```
                  INIT(variable,...)
    Cette fonction est identique à la fonction
              RESET
              ,
cependant le type est réinitialisé, c'est à dire que la

              variable
               pourra ensuite prendre n'importe quel type autorisé
(numérique ou chaîne de caractères).
    Le nombre d'arguments est quelconque.

See also :
              RESET
```

1.160 ARexx function INPUT

```
                  INPUT("nom", x, y)
Place le connecteur avec le nom spécifié, aux coordonnées x, y.
La flèche du connecteur est dirigée vers la gauche.
Si la fonction réussit, elle renvoie le numéro de l'objet, sinon 0.

L'échelle verticale et l'échelle horizontale courantes sont prises
en compte pour déterminer les dimensions de cet élément (voir
              SETSCALE
              ,

              ROTATE
              ,
```

```
               SYMMETRY
               ).

See also:
               OUTPUT
```

1.161 ARexx function JUNCTION

```
                   JUNCTION(x,y)
Place a junction at the specified coordinates. The currents
vertical and horizontal scales are used to draw it (see

               SETSCALE
               ).

If the placement is OK, this function returns the number of
the new object, else 0.
```

1.162 ARexx function LEN

```
  LEN("chaîne")
Renvoie la longueur de la chaîne de caractères passée
en argument.
```

1.163 ARexx function LIBSPATH

```
                   LIBSPATH("path")
Defines the path where the symbols files are found. This
function can modify the path specified by the
               LIBS
                 tooltype at any moment.

The function returns the path that was in use BEFORE it was called.
If the specified path is the null string, the current path is not
modified: LIBSPATH(""). Use it to know where are stored the libraries.

Example:
    'LIBSPATH("Work:AmiCAD/Symbols")'; oldpath=result
    ....
    'LIBSPATH("'oldpath'")'     /* set the initial path */
```

1.164 ARexx function LINE

```
                   LINE(object_number)
Cette fonction renvoie le numéro de ligne où est situé l'objet
spécifié.
The object number must be set beetween the values 1
```

```
and
              OBJECTS
              .

See also:
              COL
              ,
              COORDS
              .
```

1.165 ARexx function LINKREF

```
                  LINKREF(objet1,objet2)
Cette fonction permet d'affecter à un objet du type composant
une référence. L'objet déterminant la référence doit être du
type texte et ne doit pas être déjà lié à un autre composant.
The object numbers must be set beetween the values 1
and
              OBJECTS
              , they can be placed in any order.

Exemple : LINKREF(FIRSTSEL,NEXTSEL(FIRSTSEL))

See also:
              SETREF
              ,
              GETREF
              ,
              LINKVAL
              ,
              SETVAL
              ,
              GETVAL
```

1.166 ARexx function LINKVAL

```
                  LINKVAL(objet1,objet2)
Cette fonction permet d'affecter à un objet du type composant
une valeur. L'objet déterminant cette valeur doit être du
type texte et ne doit pas être déjà lié à un autre composant.
The object numbers must be set beetween the values 1
and
              OBJECTS
              , they can be placed in any order.

Exemple : LINKVAL(FIRSTSEL,NEXTSEL(FIRSTSEL))

See also:
              SETREF
              ,
              GETREF
              ,
```

LINKREF
,
SETVAL
,
GETVAL

1.167 ARexx function LOAD

LOAD("nom_fichier")
Charge le fichier schéma spécifié dans la fenêtre courante.
La fenêtre perd son contenu, même s'il avait été modifié.
Utilisez la commande
MODIF
pour savoir si le schéma a été
modifié auparavant, sans avoir été sauvé.
La fenêtre prend le nom du fichier qui a été chargé.
Renvoie 0 si tout s'est bien passé, sinon un code d'erreur.

See also:
OPEN
,
SAVE

1.168 ARexx function LOADCLIP

LOADCLIP(unité_clip,"fichier")
Charge le fichier spécifié dans l'unité spécifiée. Si le
nom du fichier n'est pas spécifié, la recherche de celui-ci
s'effectue dans le chemin courant, puis dans le chemin spécifié
par le type d'outil
CLIPS
.

Warning: le clip n'est pas pour autant placé sous le curseur
ou sur le schéma, utilisez pour cela le menu
Édition/Coller
ou la fonction
PASTE
.

See also:
CLIPUNIT
,
SAVECLIP
.

1.169 ARexx function LOADKEYS

```
                  LOADKEYS("fichier_macros")
Cette fonction permet de charger les combinaisons de touches
définies dans le fichier passé en argument. Ce fichier doit
avoir été préalablement sauvé par le menu
              Preferences/Touches/Sauver
                ou la fonction
              SAVEKEYS
              .

Exemple: LOADKEYS("Touches AmiCAD")

See also :
              MAP
              ,
              Mode direct
```

1.170 ARexx function LOADLIB

```
                  LOADLIB("nom_bibliothèque")
Cette fonction permet de charger une bibliothèque de symboles
en mémoire, elle renvoie 1 si la bibliothèque a été chargée.
Si le nom de la bibliothèque ne comporte pas de nom de
répertoire ou de "device" (DF0:, Work:, etc...), la recherche
s'effectuera dans le chemin courant puis dans le chemin
spécifié par le type d'outil LIB.

Exemple : LOADLIB("Symboles CMOS")

See also :
              REMLIB
              ,
              LIBSPATH
              .
```

1.171 ARexx function LOADPREF

```
                  LOADPREF("nom_fichier")
Charge les préférences contenues dans le fichier spécifié.
Ce fichier doit naturellement être au format convenable,
c'est à dire qu'il a dû être créé lors d'une
              sauvegarde
                des préférences.

See also:
              SAVEPREF
```

1.172 ARexx function LOCK

```
                    LOCK(x)
Verrouille la fenêtre d'indice spécifié (voir
               SELSHEET
               ). Si x vaut -1
toutes les fenêtres sont verrouillées. C'est à dire que pendant toute
l'exécution du script l'utilisateur ne peut plus avoir accès au
document: les menus ne sont plus accessibles et les opérations faites
à l'aide du clavier ou de la souris ne sont plus prises en compte.
Si un verrouillage est effectué sur une fenêtre déjà bloquée, un
compteur est incrémenté, il faut ensuite que la fenêtre soit
débloquée autant de fois qu'elle a été bloquée avant qu'elle ne
soit effectivement déverrouillée. Certaines fonctions, comme
le choix d'un fichier, la sauvegarde, le chargement, l'impression,
entraînent un verrouillage temporaire de toutes les fenêtres.
Vous avez généralement intérêt à verrouiller toutes les fenêtres,
sauf condition très particulière (LOCK(-1)). Pour verrouiller
uniquement la fenêtre active, utilisez l'écriture suivante:
    LOCK(SELSHEET(-1))

Valeur renvoyée: code supérieur ou égal à zéro si opération réussie,
    -1 si verrouillage impossible (manque de mémoire ?).

Nota: les fenêtres sont automatiquement débloquées lorsque l'exécution
d'un script se termine. S'il se produit un blocage vous pouvez
également mettre fin à cette situation en cliquant deux fois sur
l'icône du programme située sur l'écran du Workbench.
Cette fonction ne peut être appelée depuis une macro clavier (il
y aurait risque de blocage total des entrées/sorties).

See also:
               UNLOCK
               .
```

1.173 ARexx function MACRO

```
                    MACRO(x)
Exécute la macro associée à l'une des dix touches de fonctions F1 (x=1)
à F10 (x=10).   Cette  fonction est surtout pratique pour  exécuter des
appels de macros entre elles mêmes. Je l'utilise en liaison avec le
menu
               Macros/Mode~direct
               .
Ainsi la touche F4 peut par exemple appeler la macro associée à la
touche F5 (MACRO(5)).
Valeur renvoyée: résultat du calcul de la macro.

À noter que le type du résultat de la macro peut être quelconque:
numérique ou chaîne de caractères.
```

1.174 ARexx function MAP

```
                    MAP("combinaison touches", "séquence")
Cette fonction permet de programmer une macro-commande, qui
sera exécutée lors de l'appui d'une certaine combinaison de
touches. Les touches ALT, SHIFT et CTRL peuvent être utilisées
pour définir la touche (il faut d'ailleurs utiliser au moins
l'une d'entre elles).
Seules les touches définies par un seul caractère sont
définissables (pour l'instant): vous ne pouvez définir par
cette fonction, ni les touches de fonction, ni les flèches.

La séquence peut inclure n'importe quelle autre commande, elle
doit être incluse entre des parenthèses (chaîne de caractères).
La définition peut commencer par le signe =, le résultat de
la macro alors automatiquement affiché, s'il y en a un.

À noter que ces définitions sont sauvées dans le fichier de
configuration AmiCAD.keys lors d'une opération de
sauvegarde des
               Preferences
               , et sont donc récupérées
lors d'une opération de lecture ou au lancement du programme.

Exemples:
    MAP("shift-ctrl-a","SAVE(""Prog:Projets/Schémas/Essai schéma"")")
    MAP("CTRL-)","CALL(""Swap"")")

See also:

               READMAP
               ,
               UNMAP
               , définition des
               touches de fonction
               .
```

1.175 ARexx function MARK

```
                    MARK(object_number,...)
Permet de sélectionner un ou plusieurs objets.
The object numbers must be set beetween the values 1
and
               OBJECTS
               .
Cette fonction renvoie le nombre d'objets qui ont été
effectivement sélectionnés (les objets qui étaient déjà
sélectionnés ne sont pas comptabilisés).

See also:
               UNMARK
               ,
               MARKZONE
               .
```

1.176 ARexx function MARKZONE

```
                  MARKZONE(x0, x1, y0, y1)
Permet de sélectionner les objets compris dans le rectangle
défini par les points de coordonnées x0, y0 et x1, y1. Les
éléments doivent être entièrement dans la zone pour être
sélectionnés.
Cette fonction ne renvoie aucune valeur particulière.

Pour sélectionner l'ensemble des éléments du schéma, utilisez
la commande suivante:
    MARKZONE(0,0,WWIDTH(-1)-1,WHEIGHT(-1)-1)

See also:
              MARK
              ,
              UNMARK
              .
```

1.177 ARexx function MENU

```
  MENU("menu title")
The procedure associated to the menu is executed. You can specify the
only the first letters of the menu title (the first matching entry
will be called). Some menu entries can't be called (see the menus
descriptions).
Some menu titles including spaces, these spaces must be "solid spaces"
obtained with the ALT and SPACE keys. This is necessary because
AmigaGuide don't find the nodes that include some spaces (or maybe I
did an error?).

    Ex: MENU ("Quit")   Close all the documents.
  MENU ("Copy")   Copy the selected objects in the current clip.

Return: 1 if the menu entry was found, else 0.

Note: the menu title have to match the localized strings, the ARexx scripts
using this function have to be translated in foreign langages if their catalogs
exist.
    Example: MENU("Copier") in french becomes MENU("Copy") in english.
```

1.178 ARexx function MESSAGE

```
  MESSAGE("chaîne")
Affiche une requête contenant le message spécifié. Ce texte peut
contenir de une à treize lignes séparées par des sauts de lignes.
    Ex: MESSAGE("Ceci est un"+CHR(10)+"message.")

Valeur renvoyée: 0.
```

1.179 ARexx function MESURE

```
                  MESURE(dimension fenêtre)
Renvoie une des dimensions de la fenêtre courante. La valeur qui
est renvoyée dépend de celle passée en argument:
    égal 0: coordonnée gauche de l'emplacement de la fenêtre
    égal 1: coordonnée haute de l'emplacement de la fenêtre
    égal 2: largeur de la fenêtre (en pixels)
    égal 3: hauteur de la fenêtre (en pixels)
    égal 4: largeur maximale pouvant être prise par la fenêtre
    égal 5: hauteur maximale pouvant être prise par la fenêtre
    égal 6: largeur de l'écran (idem
              SWIDTH
              )
    égal 7: hauteur de l'écran (idem
              SHEIGHT
                Si la fenêtre est cachée la valeur renvoyée est toujours nulle,
sauf pour les dimensions de l'écran, nulles uniquement si l'écran
est fermé (quand toutes les fenêtres sont cachées).
Si la fenêtre est réduite la valeur déterminant sa dimension
est renvoyée sous forme négative (soit pour les valeurs de
l'argument égales de 0 à 3 compris).

See also:
              WINDOW
```

1.180 ARexx function MODIF

```
                  MODIF(fenêtre)
Permet de savoir si une fenêtre contient un schéma ayant
été modifié, sans avoir été sauvé. L'argument peut être
positif ou nul, comme renvoyé par la fonction
              SELSHEET
               :
c'est l'information concernant cette fenêtre qui est
alors renvoyée. Si l'argument est négatif, c'est pour
la fenêtre courante que l'information est renvoyée.

La valeur renvoyée est nulle si le schéma n'a pas été
modifié depuis la dernière sauvegarde, sinon elle est
normalement égale à 1.

Exemple:
    'MODIF(-1)'
    if result~=0 then 'MENU("Sauver")'

    Autre écriture (plus rapide):
    'IF(MODIF(-1),MENU("Sauver"),0)'

    Ce petit script peut être utilisé pour une sauvegarde
    automatique.
```

1.181 ARexx function MOVE

```
                  MOVE(object_number, dx, dy)
Cette fonction permet de déplacer un objet du nombre de
pixels spécifié par les valeurs de dx et dy.
The object number must be set beetween the values 1
and
              OBJECTS
              .

Aucune valeur n'est renvoyée par cette fonction.
```

1.182 ARexx function NBSHEET

```
  NBSHEET(x)
Renvoie le nombre de schémas présents en mémoire. Ce
comptage s'effectue sur tout ou partie des fenêtres,
selon la valeur de l'argument:
 égal 0:  on ne compte que les schémas cachés
 égal 1:  on ne compte que les schémas ouverts (non cachés ou réduits)
 égal -1: on compte tous les schémas

Exemple:
    NBSHEET(-1)-NBSHEET(0)  'renvoie le nombre de fenêtres réduites
```

1.183 ARexx function NEW

```
  NEW("titre")
Provoque l'ouverture d'une nouvelle fenêtre, le nom de
cette fenêtre est celui passé en argument. Si l'argument
est une chaîne nulle, la fenêtre prend le nom "Innomé".
Valeur renvoyée: 1 si réussi, 0 sinon.
```

1.184 ARexx function NEXTSEL

```
                  NEXTSEL(object_number)
Renvoie le numéro d'objet sélectionné, suivant celui qui
est spécifié en argument. Cette fonction permet ainsi,
associée à
              FIRSTSEL
              , de parcourir tous les éléments
sélectionnés.
```

1.185 ARexx function OBJECTS

```
                    OBJECTS(window)
This function returns the number of objects that are present
on the specified window, or on the current window if the
argument is -1.

 See also :
               SELSHEET
```

1.186 ARexx function OPEN

```
                    OPEN("nom_fichier")
Cette fonction permet de charger un fichier dans une nouvelle
fenêtre, mais à la différence de la fonction
               LOAD
               , elle
permet également le chargement de plusieurs fichiers. Il
suffit pour cela de spécifier des caractères génériques
(#?[]) dans le nom du fichier, comme prévu sous DOS.
Tous les schémas correspondant à ce format seront chargés.
Une nouvelle fenêtre est ouverte pour chacun des schémas
trouvés, correspondant à la demande.
La fonction renvoie 0 si tout s'est bien passé, un code d'erreur (valeur
non nulle) s'il s'est produit un problème.

Exemples :
    OPEN("Travail:AmiCAD/Schémas/Projet_TV/#?") demande le
  chargement de tous les fichiers du répertoire Projet_TV
    OPEN("#?.sch")  demande de charger tous les fichiers ayant
  l'extension .sch, situés dans le répertoire courant

See also:
               CLOSE
               ,
               LOAD
               .
```

1.187 ARexx function OUTPUT

```
                    OUTPUT(nom_connecteur, x, y)
Cette fonction permet de placer un connecteur d'entrée à
l'emplacement spécifié par les coordonnées x et y.
Elle renvoie le numéro de l'objet qui a été placé, si elle
réussit, sinon elle renvoie 0.

L'échelle verticale et l'échelle horizontale courantes sont prises
en compte pour déterminer les dimensions de cet élément (voir
               SETSCALE
               ,

               ROTATE
               ,
```

SYMMETRY
).

See also :
INPUT

1.188 ARexx function PARTNAME

PARTNAME(object_number)
Cette fonction permet de savoir quel est le nom d'un composant.
Elle renvoie une chaîne nulle si l'objet choisi n'est pas un
composant.

The object number must be set beetween the values 1
and
OBJECTS
.

See also :
PUTPART
,
GETPART

1.189 ARexx function PASTE

PASTE(clip, x, y)
Collage du contenu du tampon mémoire spécifié, à l'emplacement
donné par les arguments x et y.
Aucune valeur particulière n'est renvoyée.

See also:
COPY
,
CLIPUNIT
.

1.190 ARexx function PENWIDTH

PENWIDTH(object_number,largeur_trait)
Permet de sélectionner la largeur du trait définissant le tracé
de l'objet spécifié. La valeur du second paramètre doit être
comprise entre 1 et 255, sinon la fonction n'aura aucun effet.

Valeur renvoyée: largeur du trait utilisée AVANT l'exécution
de la fonction.

The object number must be set beetween the values 1
and
OBJECTS
.

1.191 ARexx function PICKOBJ

```
  PICKOBJ("message")
Affiche le message spécifié dans la barre de titre de la fenêtre,
puis attend un clic de l'utilisateur dans la fenêtre. Renvoie le
numéro de l'objet sur lequel a eu lieu le clic (0, s'il n'y en a pas).
L'utilisateur peut faire défiler le texte à l'aide des ascenseurs
et des boutons associés.Il peut aussi annuler l'opération en appuyant
sur le bouton droit de la souris ou sur une touche du clavier, la
valeur renvoyée est alors -1.
```

1.192 ARexx function PRINT

```
                  PRINT(rapport, rotation)

              Imprime
               le schéma, en tenant compte des arguments
rapport (facteur d'agrandissement, doit être supérieur ou
égal à 1) et rotation (si égal à 0, le schéma est imprimé
comme il est visible à l'écran, si égal à 1, le schéma est
imprimé avec une rotation de 90 degrés).
Renvoie 0 si tout se passe bien, un code d'erreur dans le
cas contraire.
```

1.193 ARexx function PUTPART

```
                  PUTPART("nom_composant", x, y)
Place le composant spécifié à l'emplacement donné par les
arguments suivants. Warning la référence et la valeur
ne sont pas placés, utilisez les fonctions
              SETREF
               et
 ou  et  pour cela.

L'échelle verticale et l'échelle horizontale courantes sont prises
en compte pour déterminer les dimensions de cet élément (voir
              SETSCALE
              ,

              ROTATE
              ,
              SYMMETRY
              ).

Cette fonction renvoie le code de l'objet qui a été placé
si elle réussit, sinon elle renvoie 0.

See also: ,
```

1.194 ARexx function READCONV

```
                    READCONV(object_number)
Cette fonction permet de savoir quel est le type de symbole
utilisé pour afficher un composant. Elle renvoie -1 si l'objet
choisi n'est pas un composant, 0 si ce composant est affiché
normalement ou 1 si c'est le symbole alterné qui est utilisé.

The object number must be set beetween the values 1
and
               OBJECTS
               .

See also : , menu Dessin/
               Alterner~symbole
```

1.195 ARexx function READDEF

```
                    READDEF("fonction")
Cette fonction permet de connaître la définition associée à
une fonction. Le nom de la fonction passée en argument doit
figurer en MAJUSCULES, entre parenthèses. Cette fonction
doit bien sûr avoir été préalablement définie à l'aide de
la fonction DEF, sinon une chaîne nulle est renvoyée.
Vous ne pouvez lire les définitions préprogrammées, comme
ABS, ASK, etc...

Exemple :
    READDEF("CIRCLE")   renvoie la définition associée à la
      fonction CIRCLE, si elle existe.

See also:

               DEF
```

1.196 ARexx function READDEV

```
                    READDEV(object_number)
Cette fonction permet de savoir quel est la porte ou le
circuit utilisé par un composant. Elle renvoie -1 si l'objet
choisi n'est pas un composant, 0 si ce composant ne comprend
qu'un circuit ou bien le numéro du circuit sélectionné (alors
supérieur ou égal à 1).

The object number must be set beetween the values 1
and
               OBJECTS
               .

See also :
               SETDEV
```

1.197 ARexx function READMAP

```
                    READMAP("key combination")
This function returns a string with the definition associated
to the argument.
If there is no definition an empty string is returned.

Examples:
    READMAP("shift-ctrl-a")
    READMAP("CTRL-i")
    READMAP("ALT-$\mathrm{\mu}$")

See also:

               MAP
               ,
               UNMAP
```

1.198 ARexx function READTEXT

```
                    READTEXT(object_number)
Renvoie le texte associé à un objet, cet objet peut être un
élément de texte, ou un connecteur (entrée ou sortie).
Si ce texte n'existe pas ou si l'objet est d'un type différent
la fonction renvoie une chaîne nulle.

The object number must be set beetween the values 1
and
               OBJECTS
               .

See also:
               SETTEXT
```

1.199 ARexx function REMLIB

```
                    REMLIB("nom_bibliothèque")
Supprime la bibliothèque spécifiée de la mémoire.
Warning : tous les éléments qui en sont issus seront
supprimés du schéma (sans avertissement).
Cette opération peut permettre de gagner de la mémoire,
en supprimant par exemple une bibliothèque qui n'est
plus utilisée.

Vous pouvez aussi utiliser le bouton "Supprimer" de la
requête de choix d'un composant.

Exemple : REMLIB("Symboles TTL")

See also :
               LOADLIB
```

1.200 ARexx function REQFILE

```
  REQFILE("titre_requête","chemin")
Ouvre la boîte de requête de fichier, avec le titre spécifié.
La requête s'ouvre en explorant le chemin spécifié. Ce chemin
doit correspondre à celui d'un répertoire ou d'un volume.

Renvoie le nom du fichier sélectionné ou une chaîne nulle
si l'utilisateur clique sur le bouton Annule.

Exemple: REQFILE("Choisissez un fichier","Work:AmiCAD")
```

1.201 ARexx function REQSHEET

```
                  REQSHEET("titre")
Cette fonction permet de choisir, à l'aide d'une boîte de requête,
un schéma parmi ceux qui sont chargés en mémoire. Le titre est
affiché en première ligne, il ne doit pas comprendre de saut de
ligne. Le numéro de schéma choisi (de 1 à x, selon le nombre de
schémas présents en mémoire) est renvoyé. Si l'utilisateur
appuie sur le bouton droit de la souris, ou bien si un problème
survient une valeur inférieure ou égale à zéro est renvoyée.
Le fonctionnement est analogue à celui obtenu après un double
clic sur le
              bouton central
               de la souris, cependant avec cette
fonction vous pouvez choisir le titre affiché en en-tête, et
le numéro du bouton choisi est renvoyé.
```

1.202 ARexx function REQUEST

```
  REQUEST("titre")
Affiche une requête avec le texte spécifié (jusqu'à treize
lignes de texte, séparés par des sauts de ligne), et deux
boutons OUI et NON. Si l'utilisateur clique sur le bouton OUI
ou appuie sur ENTRÉE, cette fonction renvoie 1.
Si l'utilisateur clique sur le bouton NON ou appuie sur la
touche ESC, la valeur renvoyée est 0.
En cas de problème (manque mémoire par exemple), la valeur
renvoyée est négative.
```

1.203 ARexx function RESET

```
                  RESET(variable,...)
    Cette fonction est identique à la fonction
              INIT
              ,
cependant le type n'est pas réinitialisé.
Si la
```

```
              variable
               est du type numérique elle prend la valeur 0, si
c'est une chaîne de caractères elle devient une chaîne vide ("").
    Le nombre d'arguments est quelconque.

Exemple : RESET(A,B,C)

See also :
              INIT
```

1.204 ARexx function ROTATE

```
                  ROTATE(object_number, rotations)
Fait tourner l'objet spécifié du nombre de quarts de tour
spécifié. Si le numéro d'objet est nul, c'est le mode de
placement courant qui est affecté: les placements effectués
ensuite tiendront compte de cette nouvelle position, que
ce soit par une macro (
              PUTPART
               par exemple) ou un menu.
Cette fonction renvoie 0 si l'opération a échoué (par
manque de place par exemple), sinon elle renvoie 1.

See also :
              SYMMETRY
              ,
              SETSCALE
```

1.205 ARexx function SAVEIFF

```
                  SAVEIFF("file name")
The current document is saved using the IFF format.
A 0 is returned if all worked.

See also:
              Project menu/Save IFF
```

1.206 ARexx function SAVE

```
                  SAVE("nom_fichier")
Sauve le document sous le nom spécifié. Si le fichier existait déjà
aucun avertissement n'a lieu. Renvoie 1 si aucun problème ne
survient, sinon 0. Un fichier .info comportant une icône est
créée si le menu
              Preferences/Sauver~icône
               est marqué, un fichier
comportant l'extension .bis est créé si le menu

              Preferences/Copie schéma sauvé
               est marqué.
```

See also:
SAVECOPY
,
SAVEICON
.

1.207 ARexx function SAVECLIP

SAVECLIP(unité,"nom_fichier")
Cette fonction permet de sauver le clip spécifié dans le fichier
spécifié. Naturellement le clip doit auparavant avoir été rempli
à l'aide de la fonction COPY, par exemple.

See also :
LOADCLIP
,
CLIPUNIT
.

1.208 ARexx function SAVECOPY

SAVECOPY(1/0/-1)
Cette fonction permet de fixer, comme le menu
Preferences/Copie schéma sauvé
,
si une copie du schéma sera créée lors d'une opération de sauvegarde.
Cette fonction renvoie 1 si le menu était marqué AVANT l'exécution
de la fonction, ou 0 s'il ne l'était pas.
Pour lire l'état du menu sans le modifier, utilisez la commande
SAVECOPY(-1).

See also:
SAVEICON
.

1.209 ARexx function SAVEICON

SAVEICON(1/0/-1)
Cette fonction permet de fixer, comme le menu
Preferences/Sauver~icône
,
si une icône sera créée lors d'une opération de sauvegarde.
Cette fonction renvoie 1 si le menu était marqué AVANT l'exécution
de la fonction, ou 0 s'il ne l'était pas.
Pour lire l'état du menu sans le modifier, utilisez la commande
SAVEICON(-1).

See also:
SAVECOPY

.

1.210 ARexx function SAVEKEYS

```
                    SAVEKEYS("nom fichier")
Cette fonction permet de sauver les combinaisons de touches alors
définies (combinaisons de touches et touches de fonction) dans le
fichier passé en argument. Ces définitions pourront ensuite être
définies par le menu
               Preferences/Touches/Charger
                ou la fonction

               LOADKEYS
               .

See also : MAP,
               Mode direct
```

1.211 ARexx function SAVEPREF

```
                    SAVEPREF("nom fichier")
Cette fonction permet de sauvegarder les préférences courantes
sous le nom spécifié. Elle renvoie 0 si tout se passe bien, un
code d'erreur dans le cas contraire

See also:
               LOADPREF
```

1.212 ARexx function SCREEN

```
                    SCREEN (mode, largeur, hauteur,"fonte",taille)
Cette fonction permet de choisir la résolution de l'écran
d'AmiCAD sans avoir à passer par la requête Asl, comme avec
le menu "Preferences/Choix mode écran". De plus vous pouvez
spécifier quelle sera la fonte utilisée pour les menus, les
boîtes de requêtes, etc... Le nom de la fonte doit être complet,
y compris l'extension ".font"
Cependant il vous faudra connaître les valeurs associées
aux divers types d'écran, un moyen simple pour les connaître
est de l'afficher avec la fonction
               SCRMODE
               , en tapant la commande
=SCRMODE dans la requête suivant l'appel au menu
               Macros/Mode direct
               .
    Ainsi les valeurs suivantes sont possibles:
    - 561188 pour un écran en super résolution entrelacée (SUPER72)
    - 561152 pour un écran en SUPER72 haute résolution.
    - 233509 pour un écran MultiScan Productivité entrelacée
    - 167936 pour un écran PAL haute résolution
```

```
    - 135168 pour un écran PAL basse résolution
La fonction renvoie le code en vigueur AVANT son exécution.

Le programme ne gère pas correctement, pour l'instant, les
dimensions supérieures à la portion visible à l'écran.

L'écran comprend toujours huit couleurs.

Exemples:
    SCREEN(561188,800,600,"courier.font",15) ' écran "normal" (Super 72)
    SCREEN(135168,320,200,"topaz.font",8)    ' écran basse résolution (pour " ↩
       zoomer")

See also:
              SCRMODE
              ,
              SHEIGHT
              ,
              SWIDTH
              ,
              FONTNAME
              ,
              FONTSIZE
              .
```

1.213 ARexx function SCRMODE

```
                  SCRMODE
Cette fonction renvoie la valeur associée au mode d'écran courant.
Elle n'a besoin d'aucun argument.

See also:
              SCREEN
```

1.214 ARexx function SECURITY

```
                  SECURITY(nombre boucles)
Cette fonction détermine le nombre maximal de boucles pouvant
être effectuées par une des fonctions
              FOR
               ou
              WHILE
              . Ceci
permet de sortir des boucles sans fin assez simplement.
La valeur par défaut est égale à 500. Vous pouvez entrer
une valeur allant de 1 jusqu'à 2^31-1 (2147483647). La
fonction renvoie la valeur en cours avant son exécution.

Vous pouvez donner à cette fonction un argument de grande
valeur sans craindre de blocage, il est effet maintenant
possible d'interrompre une boucle en appuyant simultanément
sur les touches CTRL, ALT et ESC.
```

```
Si la valeur passée en argument est nulle seule la valeur
actuelle est renvoyée, sans modification.
```

1.215 ARexx function SELECT

```
  SELECT("texte")
Cette fonction permet d'ouvrir une boîte de requête comportant
un nombre variable de boutons, comportant chacun un texte
choisi par l'utilsateur. Si l'utilisateur clique sur l'un
de ces boutons, la fonction renvoie le rang du bouton (en
commençant par la valeur 1 pour le premier, 2 pour le
second, etc...). Il peut y avoir jusqu'à 13 boutons.
Le format du texte passé en argument doit être le suivant:
    - une première ligne, affichée en titre,
    - une seconde ligne, correspondant au texte du premier bouton
    - une troisième ligne, correspondant au texte du second bouton,
    - et ainsi de suite, autant de lignes que de boutons...
Chacune des lignes est séparée des suivantes par un saut de
ligne (CHR(10)).
Si la valeur renvoyée est négative ou nulle, c'est que
l'utilisateur a appuyé sur le bouton droit ou sur la touche
Esc, ou bien que la requête n'a pu être ouverte.

Exemple:
    SELECT("Nombre de circuits ?"+CHR(10)+"10 circuits"+CHR(10)+"20 circuits"+ ↩
       CHR(10)+"À déterminer")
```

1.216 ARexx function SELFILE

```
                  SELFILE("nom_fichier")
Cette fonction permet de choisir la fenêtre active, en
spécifiant son nom. Le nom spécifié peut être le nom
complet, ou bien le nom du fichier, sans le chemin.
Si plusieurs fenêtres possèdent le même nom, la première
fenêtre trouvée est sélectionnée.
Cependant la fenêtre sélectionnée ne passe pas au
premier plan, utilisez pour cela la fonction
              WTOFRONT
              .
De même si la fenêtre est réduite ou bien cachée, elle le
restera. Ceci est utile pour lire une ligne ou quelques
caractères d'une fenêtre sans la "réveiller".

La valeur renvoyée est positive ou nulle si la fenêtre a
été trouvée (elle correspond à la valeur qu'aurait renvoyé
la fonction
              SELSHEET
              (-1)), sinon elle est négative.

    Exemples:
  SELFILE("RAM:texte")
```

```
  SELFILE("texte")
      Ces deux exemples permettent la sélection de la
      même fenêtre (RAM:texte).
```

1.217 ARexx function SELSHEET

```
                  SELSHEET(fenêtre)
Cette fonction permet de choisir la fenêtre active ou
de connaître l'indice de la fenêtre courante.
Si l'argument est positif ou nul, la fenêtre d'indice spécifié
est sélectionnée et devient la fenêtre active.
Cependant la fenêtre sélectionnée ne passe pas au
premier plan, utilisez pour cela la fonction
              WTOFRONT
              .
De même si la fenêtre est réduite ou bien cachée, elle le
restera.

La fonction renvoie toujours l'indice de la fenêtre active
avant son exécution. Si l'indice passé en argument est
négatif, seule cette valeur est renvoyée.
```

1.218 ARexx function SETCOLOR

```
  SETCOLOR(couleur,niveau_rouge,niveau_vert,niveau_bleu)
Cette fonction permet de choisir une couleur de l'écran
AmiCAD. Le nombre correspondant à cette couleur doit être
compris entre 0 et 7 compris (l'écran a huit couleurs).
Les niveaux de couleurs sont des mots long, de 32 bits,
pour spécifier du blanc, passez trois arguments égaux à
0xFFFFFFFF, pour ramener un niveau de couleur à un niveau
intermédiaire, passez un argument de valeur 0x7FFFFFFF.

Aucune valeur n'est renvoyée par cette fonction.
```

1.219 ARexx function SETDEV

```
                  SETDEV(object_number, numéro_circuit)
Cette fonction permet de choisir le numéro de circuit ou
de porte pour un composant qui en comporte plusieurs.
Elle est utile pour les circuits comportant de multiples
portes comme les circuits TTL 7400 par exemple.

The object number must be set beetween the values 1
and
              OBJECTS
              .

La fonction peut renvoyer trois résultats possibles:
    - 0 si l'objet spécifié n'est pas un composant,
```

```
    - 1 si l'opération a réussi,
    - -1 si le numéro de circuit est incorrect.

See also :
              READDEV
```

1.220 ARexx function SETGRID

```
  SETGRID(pas_grille)
Détermine la taille de la grille utilisée pour placer les
composants sur le schéma. À noter que cette fonction ne
met pas à jour la grille à l'écran, ce qui peut poser des
problèmes, utilisez le menu "Dessin/Redessiner tout" si
vous souhaitez qu'elle soit mise à jour, en utilisant par
exemple la commande suivante :
    SETGRID(15):MENU("Redessiner")

Valeur renvoyée: valeur du pas utilisée avant l'exécution de la
commande. Utilisez la commande SETGRID(0) pour connaître le pas
de la grille sans le modifier.
```

1.221 ARexx function SETPINS

```
  SETPINS(object_number,ON/OFF)
Cette fonction permet de déterminer si les numéros de broches d'un
composant seront affichés ou non.
Si le second argument est supérieur à 0, l'affichage est validé,
s'il est égal à 0, l'affichage est supprimé, s'il est inférieur
à 0 rien n'est changé.

La valeur renvoyée est égale à 1 si l'affichage était validé AVANT
l'application de la fonction, à 0 dans le cas contraire.
Si l'objet spécifié n'est pas un composant, la valeur renvoyée est
toujours égale à -1.

Exemple: SETPINS(FIRSTSEL,-1)   renvoie l'état du 1er objet sélectionné
```

1.222 ARexx function SETREF

```
                  SETREF(object_number,"référence")
La valeur spécifiée (sous forme de chaîne de caractères) est
attribuée à l'objet spécifié. Si l'objet spécifié possédait
déjà une référence, elle est remplacée.

Exemple: SETREF(FIRSTSEL,"R2")

See also :
              LINKREF
              ,
              SETVAL
```

,
LINKVAL

1.223 ARexx function SETSCALE

SETSCALE(object_number, horizontal_scale, vertical_scale)
This function is used to modify the scales of an object.

The object number must be set beetween the values 1
and
OBJECTS
. If it's set to 0, the function
sets the current scales, used to place the new objects, like
with the menu
Drawing/Place component
.

These values can also be set using the menu

Preferences/Vertical scale
and
Preferences/Horizontal scale
.
If one of the scale values is set to 0, his value will
not be changed.

This function returns 1 if she succeeded, 0 else.

1.224 ARexx function SETTEXT

SETTEXT(object_number,"chaîne")
Cette fonction permet de fixer le texte associé à un objet.
Cet objet doit être du type texte, valeur ou référence de
composant, ou bien un connecteur.
The object number must be set beetween the values 1
and
OBJECTS
.

La valeur renvoyée est égale à 1 si la fonction a réussi,
sinon elle est nulle.

See also:
READTEXT

1.225 ARexx function SETVAL

SETVAL(object_number,"valeur")
La valeur spécifiée (sous forme de chaîne de caractères) est
attribuée à l'objet spécifié. Si l'objet spécifié possédait

déjà une valeur, elle est remplacée.

Exemple: SETVAL(FIRSTSEL,"100k")

See also :
LINKVAL
,
SETREF
,
LINKREF

1.226 ARexx function SGN

SGN(nombre)
Renvoie 1 si l'argument est positif, -1 s'il est négatif,
zéro s'il est nul.

1.227 ARexx function SHEIGHT

SHEIGHT
Cette fonction renvoie la hauteur de l'écran courant.
Elle ne nécessite aucun argument.

See also:
SCREEN
,
SWIDTH
,
SCRMODE

1.228 ARexx function STOBACK

STOBACK
Cette fonction fait passer l'écran d'AmiCAD en arrière-plan.
Elle ne nécessite aucun argument.

See also :
STOFRONT

WTOBACK

WTOFRONT

1.229 ARexx function STOFRONT

```
                    STOFRONT
Cette fonction fait passer l'écran d'AmiCAD au premier plan.
Elle ne nécessite aucun argument.

See also :
               STOBACK

               WTOBACK

               WTOFRONT
```

1.230 ARexx function STR

```
  STR(nombre)
Renvoie la chaîne de caractères correspondant à un nombre.
La base utilisée pour la conversion est la base 10 (décimal).
```

1.231 ARexx function SWIDTH

```
                    SWIDTH
Cette fonction renvoie la largeur de l'écran courant.
Elle ne nécessite aucun argument.

See also:
               SCREEN
               ,
               SHEIGHT
               ,
               SCRMODE
```

1.232 ARexx function SYMMETRY

```
                    SYMMETRY(object_number, position)
Cette fonction permet de symétriser on non un objet, par
rapport à son axe vertical (ou son axe horizontal, s'il
a été tourné d'un ou trois quarts de tour).

The object number must be set beetween the values 1
and
               OBJECTS
               . Si ce numéro est nul, la fonction
modifie la façon de placer un composant (menu "Dessin/Placer
composant").

L'argument position peut prendre trois valeurs :
    - égal à 1: la symétrie est réalisée,
    - égal à 0: pas de symétrie,
    - égal à -1: inversion de la position.
```

```
La valeur renvoyée est égale à 1 s'il y a eu une modification
de la position de l'objet, 0 dans le cas contraire.

See also :
              ROTATE
              ,
              SETSCALE
```

1.233 ARexx function TEST

```
                  TEST(object_number)
Permet de savoir si un objet est sélectionné ou non.
The object number must be set beetween the values 1
and
              OBJECTS
              .

La fonction renvoie 1 si l'objet est sélectionné, 0
dans le cas contraire.
```

1.234 ARexx function TIME

```
                  TIME(secondes)
Renvoie l'heure courante, si l'argument est non nul, les
secondes sont comprises.

Exemples:
    TIME(0)     renvoie 23:24
    TIME(1)     renvoie 23:24:27

See also:
              DATE
```

1.235 ARexx function TITLE

```
  TITLE("titre")
Spécifie le titre de la fenêtre courante. Si l'argument
est une chaîne nulle, le titre reprend son état normal,
c'est à dire que le nom du fichier y réapparaît.
Cette fonction permet de signaler une opération un peu
longue en prévenant l'utilisateur, sans pour autant
bloquer la fenêtre ni le programme.

Exemple de script:
    'TITLE("Traitement en cours...")'
    ...     traitement...
    'TITLE("")'         on remet le titre normal
```

1.236 ARexx function TXHEIGHT

```
                  TXHEIGHT("texte")
Renvoie la hauteur du texte spécifié, en pixels. Cette fonction prend
en compte l'échelle courante, ainsi que le mode de placement (rotation
éventuelle). Dans tous les cas c'est l'espace vertical occupé qui
est renvoyé.

See also:
              TXWIDTH
```

1.237 ARexx function TXWIDTH

```
                  TXWIDTH("texte")
Renvoie la largeur du texte spécifié, en pixels. Cette fonction prend
en compte l'échelle courante, ainsi que le mode de placement (rotation
éventuelle). Dans tous les cas c'est l'espace horizontal occupé qui
est renvoyé.

Cette fonction est utile pour centrer un texte dans un rectangle:
    options results
    texte="Essai texte"
    xx=100;y=100
    'TXWIDTH("'texte'")'
    l=result
    'TXHEIGHT("'texte'")'
    h=result
    'DRAW('xx','y','xx+l','y'):DRAW('xx+l','y','xx+l','y+h')'
    'DRAW('xx+l','y+h','xx','y+h'):DRAW('xx','y+h','xx','y')'
    'WRITE("'texte'",'xx','y+h')'

See also:
              TXHEIGHT
```

1.238 ARexx function TYPE

```
                  TYPE(object_number)
Cette fonction permet de connaître la nature d'un objet.
The object number must be set beetween the values 1
and
              OBJECTS
              .
La valeur renvoyée dépend du type du caractère:
    - égale à 1: c'est un composant
    - égale à 2: c'est un fil de liaison normal
    - égale à 3: c'est un arc de cercle
    - égale à 4: c'est un texte
    - égale à 5: c'est une référence de composant
    - égale à 6: c'est une valeur de composant (ou son type)
    - égale à 7: c'est une connexion
    - égale à 8: c'est un trait en pointillés
    - égale à 9: c'est un bus
```

```
    - égale à 10: c'est une ellipse
    - égale à 11: c'est un connecteur d'entrée
    - égale à 12: c'est un connecteur de sortie
    - égale à 15: c'est un trait double
    - égale à 21: c'est une ligne personnalisée (largeur quelconque)

Vous pouvez ainsi définir des fonctions permettant de reconnaître
les objets :
    DEF COMPOSANT(O) = IF (TYPE(O) == 1, 1, 0)
    DEF FIL(O) = IF (TYPE(O) == 2, 1, 0)
    DEF CONNECTEUR(O) = IF ((TYPE(O)==11) | (TYPE(O)==12), 1 ,0)
Chacune de ces fonctions renvoie 1 si le caractère est du
type testé, sinon elles renvoient 0.
```

1.239 ARexx function UNLINK

```
                  UNLINK(object_number)
The object number must be set beetween the values 1
and
              OBJECTS
              .

Cette fonction permet de briser les liens existant entre
un composant et sa référence, ainsi que sa valeur. Si le
composant possédait ces deux éléments, la fonction renvoie
2, s'il n'en possédait qu'un seul des deux, elle renvoie 1
sinon elle renvoie 0.
Si l'objet spécifié n'est pas un composant la valeur renvoyée
est égale à -1.

Exemples : UNLINK(FIRSTSEL)
     UNLINK(PICKOBJ("Cliquez sur un composant"))

See also :
              LINKVAL
              ,
              LINKREF
```

1.240 ARexx function UNLOCK

```
                  UNLOCK(fenêtre)
Annule le verrouillage d'une fenêtre, réalisé par la
fonction
              LOCK
              .
Si l'argument est positif ou nul, seule la fenêtre
possédant l'indice spécifié est déverrouillée, si
cet argument est négatif (-1), toutes les fenêtres
sont deverrouillées.

See also:
              SELSHEET
```

.

1.241 ARexx function UNMAP

```
                    UNMAP("key combination")
This function deletes a macro key programmation, associated with
a key combination. The ALT, SHIFT and CTRL can be used to define
the combination (one of them must be used).

Examples:
    UNMAP("shift-ctrl-a")
    UNMAP("ctrl-shift-a")
    UNMAP("CTRL-)")
    UNMAP("ALT-$\mathrm{\mu}$")

See also:

               MAP
               ,
               READMAP
```

1.242 ARexx function UNMARK

```
                    UNMARK(object_number,...)
Annule le marquage des objets spécifiés.

The object numbers must be set beetween the values 1
and
               OBJECTS
               .

Renvoie le nombre de sélections ayant été annulées.

See also:

               MARK
               ,
               MARKZONE
```

1.243 ARexx function VAL

```
                    VAL("chaîne")
Renvoie la valeur numérique correspondant à la chaîne de
caractères passée en argument. Seuls les caractères
correspondant à des nombres entiers sont pris en compte.
Le nombre doit figurer sous forme décimale, sauf s'il est
précédé du préfixe $, qui signale qu'il s'agit d'un
nombre sous forme hexadécimale.

Exemples:
```

```
    VAL("14")       renvoie 14
    VAL("14.3")     renvoie 14
    VAL("$5F")      renvoie 95 (le $ spécifie une chaîne hexadécimale)

See also :
              STR
```

1.244 ARexx function VERSION

```
  VERSION(arg)
This function returns the current version number of the program,
the processor it was compiled for or a copyright message, depending
on the argument.
This function can be used to test if a program can execute or
not a script.

Returned values (results can differ with the program version number):
    VERSION(0)      returns  1.3
    VERSION(1)      returns 68000, 68020 or 68060
    VERSION(2)      returns "© R.FLORAC  1er juin 1998"

Example of script:
    'VERSION(0)'
    if result < 1.3 then do
  'MESSAGE("This program version"+CHR(10)+"can''t do this job!")'
  exit
    end
    ...
```

1.245 ARexx function VSCALE

```
                  VSCALE(object_number)
This function returns the vertical scale value of
the specified object.
The object number must be set beetween the values
1 and
              OBJECTS
              .

See also:
              HSCALE
              ,
              SETSCALE
              .
```

1.246 ARexx function WHEIGHT

```
                  WHEIGHT(code_fenêtre)
Renvoie la hauteur utile de la fenêtre spécifiée. Le code de
la fenêtre correspond à la valeur renvoyée par la fonction
```

```
              SELSHEET
              .
Si l'argument vaut -1, c'est la fenêtre courante qui est traitée.
La valeur renvoyée correspond en fait à la largeur du document,
en pixels.

Rappel: les fenêtres sont du type SuperBitmap, ce qui veut dire
que leur surface visible n'est pas forcément égale à celle qu'elles
pourraient prendre, si elles sont plus grandes que l'écran.

See also :
              WWIDTH
              ,
              MESURE
              ,
              WINDOW
              ,
              SCREEN
              ,
              DIMSHEET
```

1.247 ARexx function WHILE

```
                  WHILE(fin,action1,...)
Cette fonction est identique à la fonction
              FOR
              , il
lui manque juste le premier argument. L'initialisation des

              variables
               ou des cellules testées aura donc due être effectuée
auparavant.

See also:
              SECURITY
```

1.248 ARexx function WIDTH

```
                  WIDTH(object_number)
Returns the width of the specified object, in pixels.

The object number must be set beetween the values 1
and
              OBJECTS
              .

See also:
              HEIGHT
```

1.249 ARexx function WINDOW

```
                  WINDOW(x, y, largeur, hauteur)
Cette fonction permet de redimensionner la fenêtre courante, ou
de la déplacer.
Les deux premiers arguments correspondent aux coordonnées du coin
supérieur gauche de la fenêtre dans l'écran. Les deux arguments
suivants déterminent ses dimensions.
Warning: cette fonction ne fonctionne que pour les fenêtres
"normales".Pour les fenêtres réduites, seules les coordonnées
x et y peuvent être modifiées (pour déplacer les fenêtres).
Les fenêtres cachées ne peuvent naturellement pas être modifiées,
mais il suffit de spécifier la commande WINDOW(-1,-1,-1,-1) pour
permettre leur réouverture.
Si vous ne voulez pas modifier un ou plusieurs de ces paramètres
et que vous ne connaissiez pas leur valeur, donnez-leur une valeur
négative (-1). Si les arguments ont des valeurs trop grandes ou
aberrantes, le programme essaiera d'y remédier au mieux.
La valeur renvoyée est égale au nombre de valeurs ayant été prises
en compte.

Exemples:
    WINDOW(0,0,-1,-1)   déplace la fenêtre en haut à gauche de l'écran
      sans modifier ses dimensions.
    WINDOW(-1,-1,200,50) change les dimensions de la fenêtre

See also:
              MESURE
              ,
              type d'outil WINDOW
```

1.250 ARexx function WRITE

```
                  WRITE("string", x, y)
Place a text at the specified coordinates.

The current horizontal and vertical scales are used (see

              SETSCALE
              ,
              ROTATE
              ,
              SYMMETRY
              ).

If the placement is OK, this function returns the number of
the new object, else 0.

See also:
              READTEXT
              .
```

1.251 ARexx function WTOBACK

```
                  WTOBACK(fenêtre)
Envoie la fenêtre d'indice spécifié ou la fenêtre
courante (argument égal à -1) en arrière-plan.

See also:
              SELSHEET
              ,
              WTOFRONT
              ,
              STOBACK
              ,
              STOFRONT
```

1.252 ARexx function WTOFRONT

```
                  WTOFRONT(fenêtre)
Envoie la fenêtre d'indice spécifié ou la fenêtre
courante (argument égal à -1) à l'avant-plan.

See also:
              SELSHEET
              ,
              WTOBACK
```

1.253 ARexx function WWIDTH

```
                  WWIDTH(code_fenêtre)
Renvoie la largeur utile de la fenêtre spécifiée. Le code de
la fenêtre correspond à la valeur renvoyée par la fonction
              SELSHEET
              .
Si l'argument vaut -1, c'est la fenêtre courante qui est traitée.
La valeur renvoyée correspond en fait à la hauteur du document,
en pixels.

Rappel: les fenêtres sont du type SuperBitmap, ce qui veut dire
que leur surface visible n'est pas forcément égale à celle qu'elles
pourraient prendre, si elles sont plus grandes que l'écran.

See also :
              WHEIGHT
              ,
              MESURE
              ,
              WINDOW
              ,
              SCREEN
              ,
              DIMSHEET
```

1.254 Use of the keyboard

HELP Lance le programme AmigaGuide, celui-ci charge le fichier d'aide AmiCAD.guide. Ce fichier guide doit se trouver dans le répertoire où est situé le programme AmiCAD, ou bien à l'endroit spécifié par le type d'outil HELPFILE .

Edit commands

DEL Effacement du ou des objets sélectionnés

FLÈCHES déplacement curseur

FLÈCHES DE DÉPLACEMENT (GAUCHE / DROITE / HAUT / BAS) : déplacement des éléments sélectionnés, pixel par pixel ou par 10 pixels si la touche SHIFT est aussi appuyée.

ALT FLÈCHE: Édition des rayons de l'arc sélectionné, pixel par pixel ou par 10 pixels si la touche SHIFT est aussi appuyée.

CONTROL FLÈCHE: Édition des angles de l'arc sélectionné, degré par degré ou par par variation de 10 degrés si la touche SHIFT est aussi appuyée.

CTRL : permet de désélectionner des éléments, si utilisée en même temps que le bouton gauche de la souris lors de la sélection.

TAB : copie (clonage) des éléments sélectionnés

ALT HELP : affichage succinct du rôle des touches ALT avec les touches de fonctions.
ALT F1/F10 : programmation d'une macro-commande, puis répétition.
SHIFT ALT F1/F10: reprogrammation des macro-commandes.

Utilisation des touches de FONCTION

F3 : ouverture d'une nouvelle fenêtre, choix du fichier à charger
F4 : ouverture d'une nouvelle fenêtre
F9 : déplacement de l'écran en arrière
F10 : déplacement de la fenêtre en arrière

1.255 Utilisation des touches de fonction

Les dix touches de fonction peuvent être associées à des séquences de commandes comprenant les appels aux fonctions que vous aurez définis. Ces séquences programmées sont sauvegardées dans le fichier "s:AmiCAD.keys", à l'aide du menu

```
              Preferences/Touches/Sauver
              . Celles-ci seront alors
  rechargées lors de chaque lancement du programme.
L'appel d'une combinaison se fait en appuyant sur une touche ALT et sur
  la touche de fonction voulue. La redéfinition de cette touche peut se faire
  en appuyant sur les touches SHIFT et ALT avant d'appuyer sur la touche de
  fonction à redéfinir.

Exemple:
    - soit à programmer la séquence
              SETGRID
              (5) sur la touche F1,
  a) appuyez sur ALT, SHIFT et enfin F1,
  b) une requête est ouverte, saisissez-y la macro:
      SETGRID(5)
  c) validez, la définition est sauvée en mémoire,
  d) appuyez maintenant les touches ALT et F1, la macro est
  exécutée.
  Pour la redéfinir ultérieurement, recommencez en a).
```

1.256 bgui.library

```
  L'auteur de la bibliothèque bgui.library est Jan van den Baard.
  Cette bibliothèque est disponible dans le domaine public (Sites Aminet
  par exemple).

      BGUI release 1.1
      (C) Copyright 1993-1994  Jaba Development
      (C) Copyright 1993-1994 Jan van den Baard
       Écrit avec le compilateur DICE v3.0 par

    Poste: Jan van den Baard
           Bakkerstraat  176
           3082 HE Rotterdam
          Holland

       Modem: 2:286/407.53  (Jan van.den.Baard)
       EMail: jaba@grafix.wlink.nl
```

1.257 Aide en ligne

```
                  Une aide en ligne peut être obtenue à tout moment, à l'aide du
  fichier AmiCAD.guide. Ce fichier doit se situer dans le même
  répertoire que le programme ou bien à un emplacement et
  un nom spécifiés dans le type d'outil
              HELPFILE
               de l'icône du
  programme (cette information est prise en compte lors du
  lancement du programme).

  Pour lancer l'aide vous pouvez appuyer sur la touche HELP,
  à tout moment ou bien lors de la sélection d'un menu, ce qui
```

vous permet d'avoir directement l'aide concernant ce menu.
Vous pouvez aussi utiliser le menu
Projet/Aide
et donner le nom d'un noeud (node) du fichier d'aide. Vous pouvez ainsi trouver une aide très rapidement pour n'importe quelle fonction, en donnant son nom.

À noter que vous pouvez également provoquer l'apparition de l'aide sur une fonction donnée, si une erreur survient, provoquée par cette fonction. Appuyez simplement sur la touche HELP alors que la requête signalant l'erreur est encore affichée.

1.258 Quelques

Les définitions suivantes peuvent être intégrées au fichier AmiCAD.keys, soit à l'aide d'un éditeur de texte, soit en les définissant sous AmiCAD (voir fonction
MAP
) puis en choisissant le menu
Preferences/Keys/Save
.

Macro permettant de choisir un composant en tapant uniquement les premières lettres de son nom (voire seulement la première):

```
IF((PP=ASK("Composant?"))<>"",GETPART(PP):MENU("Placer"),0)
```

Macro permettant d'appeler une fonction associée à une autre macro, ceci pour tous les objets sélectionnés (ici c'est MACRO(5) qui est appelée, soit la macro associée à la touche F5):

```
O=FIRSTSEL:WHILE(O,MACRO(5):O=NEXTSEL(O))
```

Sélection de tous les éléments du document:

```
MARKZONE(0,0,WWIDTH(-1)-1,WHEIGHT(-1)-1)
```

Chargement d'une bibliothèque de composants à la demande, sans passer par la requête bgui:

```
LOADLIB(ASK("Bibliothèque à charger?"))
ou mieux:
IF((LIB=ASK("Bibliothèque à charger?"))<>"",LOADLIB(LIB),0)
```

Ajout du signe "ohm" à la valeur d'une résistance (la VALEUR de la résistance en question doit être seule sélectionnée):

```
SETTEXT(FIRSTSEL,READTEXT(FIRSTSEL)+CHR(139))
```

Chargement d'un clip, passage de ce clip sous le curseur:

```
LOADCLIP(1,"Additionneur"):MENU("Coller")
```

Pour lier un texte à un composant (en tant que valeur ou référence):

```
LINKVAL(FIRSTSEL,PICKOBJ("Cliquez sur la valeur de ce composant"))
LINKREF(FIRSTSEL,PICKOBJ("Cliquez sur la référence de ce composant"))
```

1.259 BUG(s) ? (mais oui ! sûrement... (malheureusement !))

Ce programme m'a demandé beaucoup de travail aussi je vous demanderais de bien vouloir être indulgent quant aux erreurs toujours possibles survenant lors de son utilisation. Je vous saurais gré de bien vouloir
m'avertir
si vous constatez une ou des anomalies.

L'utilisation du programme sous un écran CyberGraphics peut donner des résultats quelquefois bizarres (en fait la fonction de dessin en mode COMPLEMENT ne fonctionne alors pas ou très mal, pour les remplissages de surfaces, alors que le même programme fonctionne parfaitement sous les modes AGA).

Il peut arriver qu'il survienne une erreur système lors de l'exécution d'un script ARexx, si une commande provoque une erreur. Ce problème est normalement évité en suivant la structure donnée en exemple dans le script squelette.AmiCAD pour les scripts ARexx.

L'écran AmiCAD est un écran public, ainsi vous pouvez ainsi ouvrir une (ou même plusieurs) autre(s) application(s) sur son écran. Cependant, fermez toujours ces autre fenêtres AVANT de quitter AmiCAD, sinon il peut y avoir quelques problèmes...

1.260 Historique

Version 1.3 2 mai 1998
Correction du bug de la fonction "Restaurer" (enfin... ?)
Corrections bugs traitement boîtes (impression et clips)...
Correction bug fonction
SETPINS
. Ajout fonction
BOX
.
Modification des fonctions
WRITE
,
INPUT
et
OUTPUT
.

Version 1.2 12 avril 1998
Corrections bugs : fonction DATE, utilisation fonctions utilisateur quelquefois défaillantes (DEF), fonctions SETDEV, SETGRID, REMLIB...
Ajout fonctions SAVEALL et GETCOLOR.
Amélioration de la gestion du menu Symétrie. Ajout du traitement des opérations de rotation, symétrie, agrandissement et diminution sur le clip en cours.
Passage de l'écran de 8 à 16 couleurs: les éléments sont maintenant dessinés de couleurs différentes, selon leur type.
Ajout de l'objet rectangle (pas de pointillés pour l'instant).
Ajout d'un menu dans les préférences pour garder ou non le nom complet

```
  du fichier dans la barre de titre. Modification des indications
  portées dans la barre de titre (suggestion de Sébastien VEYRIN-FORRER).
  Amélioration du script Grille (création d'une grille dans le rectangle
  défini par l'utilisateur).
  Ajout du catalogue italien (par Massimo Basso), nouvelle version
  du catalogue espagnol (Benjamin Morente).
  Localisation des bibliothèques de symboles.

Version 1.1 8 mars 1998
  Remplacement du fichier de configuration Configuration.AmiCAD par
  le fichier AmiCAD.prefs (plus besoin d'utiliser la ConfigFile.library,
  cependant les nouveaux fichiers de configuration ne sont pas
  compatibles avec les anciens).
  Amélioration du script d'installation.
  Écriture du catalogue allemand (par Henk Joans).
  Modification ouverture écran (écran de dimensions égales à celui du
  Workbench par défaut).
  Correction de quelques petits bugs (largeur des boutons dans les
  requêtes, largeur des fenêtres réduites, gestion macro ARexx SYMMETRY)
  ainsi que de quelques autres plus importants (macro
               LOADCLIP
               ).
  Ajout tooltypes
               SHEET_WIDTH
                et
               SHEET_HEIGHT
               .
  Quelques optimisations du code (programme plus court).
  Ajout édition largeur trait composants, textes, ellipses, arcs, etc.
  Modification gestion des flèches curseur pour déplacement des objets
  et édition arcs et ellipses. Modification de la gestion des groupes.
  Ajout de la fonction ARexx
               SETPINS
               .
  Suppression de l'utilisation de la ConfigFile.library, remplacée par
  des routines internes, plus courtes.
  Correction bug couleurs écran AGA.
  Suppression des "requesters", remplacés par des fenêtres...

Version 1.00  18 janvier 1998
  Première version, écrite pour système 3.0. 1ère diffusion sur Aminet.
```

1.261 Help-me!!

```
                 This program can be very enhanced. But I have not all the docs
    that I need, and not such time. So, if you can send me some docs or
    do a part of the translation of this guide,
               mail me!
               .

  For the moment I'm looking for documentation on the next points:
  - description of the EPSF format (for saving under this format,
    to load the sheets with a program like ProPage or another one)...
  - routines of "clipping", to draw an object even it's not
    completly in the window (lines, but also circles, arcs, filled areas...)
```

- description of the "printer.device" (how to know the number of
  pixels that can be set by the current printer on a line, using the
  Amiga preferences).
- how to use vector fonts (CompuGraphics or Adobe...) like ProPage or
  WordWorth,
- etc...

Thanks for your suggests and collaboration.

1.262 AMÉLIORATIONS POSSIBLES

Les améliorations suivantes seront faites si le besoin s'en fait sentir et si j'en ai le temps (et aussi si j'en ai le courage !) :
- choix de l'icône créée par le programme lors d'une sauvegarde,
- amélioration des messages d'erreurs, souvent imprécis,
- amélioration de la fonction MAP (visualisation et choix des macros déjà définies),
- marquage d'emplacements dans le schéma pour les retrouver rapidement (pratique dans un schéma de grande taille),
- requête permettant le choix d'un objet quand il y a superposition,
- écriture d'une fonction zoom interne (j'utilise le freeware Lupe de Frank Toepper),
- écriture d'un éditeur de bibliothèque,
- et plus encore... (voir
  HELP!
)

Si vous souhaitez traduire cette documentation ou bien le fichier catalogue, merci de me les communiquer afin qu'ils soient distribués avec le programme.

1.263 Translation

The program was written in english. I have also written
the french catalog.

The deutsch catalog has been written by Henk Jonas:
    E-mail: subvcbhd@calvados.zrz.TU-Berlin.DE

The czech catalog has been written by Vit Sindlar:
    E-mail: SINDLAR@jackal.cis.vutbr.cz

The spanish catalog has been written by Benjamin Morente:
    E-mail: ackman@mx3.redestb.es

The italian catalog has been written by Massimo Basso:
    E-mail: cralex@amiga.dei.unipd.it

The slovenian catalog has been written by Daniel Krstic:
    E-mail: danny.k@www.comtron.si

If you want to translate the catalog in another language,
please send me it. Join the sources if possible.

I am looking for guys that could help me to translate the guide
file in english (or deutsch?). Even some translations of some
items would be appreciated.
Warning: the name of the nodes can't be changed: the program
uses them for the AmigaGuide on-line help.

1.264 AmiCAD2META

AmiCAD2META est un petit programme permettant de transformer un
fichier au format AmiCAD (c'est à dire sauvé à l'aide de AmiCAD) en
un format spécial utilisé par le programme MetaView et la bibliothèque
amigametaformat.library. Ces deux programmes ont été écrits par Henk
Jonas et permettent de sauver les fichiers AMF aux formats vectoriels
supportés par cette bibliothèque. Pour l'instant les formats suivants
sont supportés:
- WMF
- DR2D
- CGM
- GEM
- EPS
- AI
- HPGL
- ILBM

Il vous faudra obtenir ces programmes sur Aminet pour pouvoir les
utiliser (gfx/conv/MetaView, util/dtype/DT_MetaView, util/libs/amf_library),
ceux-ci n'étant pas distribués avec AmiCAD. Vous pouvez aussi écrire à
Henk Jonas pour avoir des précisions:
E-mail: subvcbhd@calvados.zrz.TU-Berlin.DE

Pour utiliser AmiCAD2Meta vous pouvez utiliser le script ARexx
Conv2Meta ou bien directement appeler AmiCAD2Meta dans une fenêtre Shell,
la syntaxe d'appel de ce programme est la suivante:

AmiCAD2Meta FROM/A,TO/K,FORCE/S,LIBS/K,VERBOSE/S,QUIET/S,PENWIDTH/N

Le premier argument est obligatoire (FROM), il spécifie quel est le fichier
à traiter. Il doit s'agir d'un fichier préalablement créé lors d'une ↩
sauvegarde
par AmiCAD. Le mot clé FROM peut être omis.
Exemples: AmiCAD2Meta Work:AmiCAD/Schémas/Logo
AmiCAD2Meta FROM Work:AmiCAD/Schémas/Logo

Le second argument spécifie le nom du fichier AMF destination (au format
Meta). S'il n'est pas donné aucune conversion ne se fera. Cela peut cependant
être utile pour vérifier un fichier. Si ce fichier destination existe déjà,
utilisez l'option FORCE pour qu'il soit ré-écrit.
Exemple: AmiCAD2Meta ... TO ... FORCE

L'argument LIBS permet de préciser où sont situées les bibliothèques de
symboles utilisées par AmiCAD.
Exemple: AmiCAD2Meta ... LIBS Work:AmiCAD/Bibliothèques

```
L'argument VERBOSE permet d'afficher un certain nombre d'informations à
  l'écran alors que l'argument QUIET permet de n'avoir aucun affichage (utile
  dans un script ARexx). Ces deux derniers arguments doivent être naturellement
  être utilisés l'un sans l'autre.
Exemple: AmiCAD2Meta ... VERBOSE

Le dernier argument (PENWIDTH) permet d'élargir (éventuellement) les traits
  du multiple spécifié. Si cet argument n'est pas donné les traits sont sauvés ↩
     avec
  la largeur présente sur le schéma, sinon leur largeur est multipliée par la
  valeur de cet argument. De nombreux programmes ne traitent pas la largeur des
  traits mais les tracent toujours de la même largeur (Wordworth sous les ↩
     formats
  CGM ou GEM, AI avec ProPage).
Exemple: AmiCAD2Meta ... PENWIDTH=2

Exemple d'appel:
    AmiCAD2Meta Work:AmiCAD/Schémas/Essai TO Work:MetaView/AmiCAD/Essai.amf LIBS ↩
        Work:AmiCAD/Bibliothèques
    AmiCAD2Meta Work:AmiCAD/Schémas/Essai VERBOSE
```

1.265 The author

```
  To contact me:

FLORAC Roland
6 Rue des Chardonnerets
Chez Corbin
17610 Chaniers
France
Phone: 05 46 93 95 71

E-mail: Roland.Florac@wanadoo.fr
```

1.266 Index

FILENAME

FILEPART

FINDPART

FINDREF

FINDVAL

FIRSTSEL

FONTNAME

FONTSIZE

FOR

Future
G

GETPART

GETREF

GETVAL
H

HEIGHT

HELP

HELPFILE

HISTORIQUE

HSCALE
I

IF

INIT

INPUT

Installation of the program
J

JUNCTION
K

Keys
L

LEN

LIBS

LIBSPATH

LINE

LINKREF

LINKVAL

LOAD

LOADCLIP

LOADKEYS

LOADLIB

LOADPREF

LOCK
M

MACRO

MACRO tooltype

Macro-command

MAP

MARK

MARKZONE

MENU

Menu Draw

Menu Edition

Menu Macros

Menu Project

Menu Preferences

Menus

MESSAGE

MESURE

MODIF

MOVE
N

NBSHEET

NEW

NEXTSEL

Numbers

Numeric values
O

OBJECTS

On-line help

OPEN

OUTPUT
P

Palette

PARTNAME

PASTE

PENWIDTH

PICKOBJ

Port ARexx

PRINT

PUTPART
R

READCONV

READDEF

READDEV

READMAP

READTEXT

REMLIB

REQFILE

REQSHEET

REQUEST

RESET

ROTATE

Running the program
S

SAVEIFF

SAVE

SAVECLIP

SAVECOPY

SAVEICON

SAVEKEYS

SAVEPREF

SCREEN

Scripts

SCRMODE

SECURITY

SELECT

SELFILE

SELSHEET

SETCOLOR

SETDEV

SETGRID

SETPINS

SETREF

SETSCALE

SETTABS

SETTEXT

SETVAL

SGN

SHEET_HEIGHT

SHEET_WIDTH

SHEIGHT

STARTUP

STOBACK

STOFRONT

STR

SWIDTH

SYMMETRY
T

TEST

Thematic ARexx functions

TIME

TITLE

TXHEIGHT

TXWIDTH

TYPE
U

UNLINK

UNLOCK

UNMAP

UNMARK
V

VAL

Variables

VERSION

VSCALE
W

WHEIGHT

WHILE

WIDTH

WINDOW

WINDOW tooltype

WRITE

WTOBACK

WTOFRONT

WWIDTH
X

X_ICON
Y

Y_ICON