AmiCAD

FLORAC Roland

COLLABORATORS			
	TITLE : AmiCAD		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	FLORAC Roland	July 31, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmiCAD

1.1 Sommaire

1.2 Distribution

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

You can reach me by sending your E-mails on the net.

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. It's ARexx interface gives it more than 130 commands, user defined functions are also available. It can use it's own variables (integer maths and strings).
The bgui.library version 39+ can be used for loading and looking for symbols, although it's not necessary for running the program. This library is not included in the package, you can get it on Aminet. The program will work perfectly without it, but the loading of libraries will be a lot harder, and the choice of the components too...

The program can work with many windows at the same time, as long as memory is available. 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. The Installer program is required for this to work.
The preference files AmiCAD.prefs and AmiCAD.keys must be copied into the same drawer as the AmiCAD program.
The AmiCAD.guide help file can be copied anywhere you like, you only have to set the AmiCAD tooltype HELPFILE to the corresponding path (HELP: for example).

WARNING: the program can't be run 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 detaches from the console, so the "Run" command is not
  required.

  AmigaDOS wildcards can be used:

AmiCAD #?.sheet   ==> all files with the extension ".sheet"
    will be loaded into individual windows.

  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 the symbol libraries are stored:

Example:
    AmiCAD LIBS Work:AmiCAD/Symbols

  The keyword CLIPS gives the path where the clips are stored:

Example:
    AmiCAD CLIPS Work:AmiCAD/Clips

  The keyword STARTUP gives the program the name of an ARexx script
  that will be run 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 SHEET_WIDTH 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 these 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 here:
WINDOW ,
HELPFILE ,
STARTUP ,
LIBS ,
MACRO ,
CLIPS ,
X_ICON ,
Y_ICON ,
SHEET_WIDTH ,
SHEET_HEIGHT,
GRIDSIZE .

Use the Workbench Information menuitem to display and modify them.

When the program is running, all the requesters that are open 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 indicators on the titlebars of the windows:
- The name of the file associated to the window (with the complete path or not),
- some letters between two braces []:,
  - a + sign if the document has been modified
  - the letter G if snap on grid is valid,
  - the letter R if components are placed with their Reference,
  - the letter V if components are placed with their Value,
  - the letter N if 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 space 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 location
  of the window that is opened when the program is run. 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 between the fields, just a ",".

  Note: if you load a file at startup, this tooltype will be
  ignored, the data will be taken from 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 run 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 sub-directory, in the AmiCAD directory.

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

See also: LIBSPATH
```

1.11 The CLIPS tooltype

```
  This tooltype gives the path where the clips are found.
  The default path is the Clips sub-directory, in the AmiCAD directory.

Examples:
    CLIPS=Work:AmiCAD/Clips
```

```
    CLIPS=User_symbols

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("Symbols special")
```

1.13 The X_ICON tooltype

```
  This tooltype is used to locate the AmiCAD AppIcon 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 locate the AmiCAD AppIcon 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 run (default value). The window is a SuperBitmap
  window, all the objects will be drawn in it.  Big values require a
  large amount 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 height of the document that will be
  opened when the program is run.

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 always 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 it's content had not been saved, a requester will ask you if you want to save 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 in the window.

Keyboard shortcut: AMIGA-S (save) or S

See also: SAVE.

ARexx call: MENU("Save~file") Warning: solid space between (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 filename.

If the file already exists a requester will ask you to confirm that you want to overwrite it.

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

See also: SAVE.

ARexx call: MENU("Save~as") Warning: solid space between (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 into 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 files are saved using only two colours (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 between (ALT SPACE) the words Save and IFF.

1.24 Project menu/Rename

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

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 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, 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 between (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 useful when you work with multiple windows.

You can place the iconified window whereever you want, when the window is iconified 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 (icon) 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 also be used by selecting an iconified window and using the key.

See also: NEW, OPEN.

ARexx call: MENU("Other~window") Warning: solid space between (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 document

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 with a value of 1, a pixel on screen will be equivalent to a dot on the paper, if you reply by 2, the document will be printed twice the size. 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 a lot of CHIP memory.

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

The windows are iconified during this process to make more CHIP memory available. If you have any problem, try to close some windows or screens not in use at the 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 information :
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 useful because you can select any window without the mouse but with the keyboard, simply by pressing an 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 (always with an ALT key) to go to the previous or 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 way 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 it's 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 it's name (Ex: COPY, PASTE...)
You can also have help for each menu entry by selecting the 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 open, 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 were modified and not saved, a requester will ask you to continue or not. The program frees all memory it was using when everything is closed.

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 save requesters will be displayed.

1.36 AmiCAD AppIcon

When AmiCAD is running, an AppIcon is placed on the Workbench screen. It's name is AmiCAD. It's purpose is to load any document by dragging it's icon onto this AppIcon. Simple, no?

This AppIcon has another function: clicking twice on it brings the AmiCAD screen to the front. 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 bad usage 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 later. Its purpose is to have a better memory handler, it's faster and 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 space the objects on the document. The default size is 10.
  The components defined in the symbol 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 between
   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 wherever you want,
  but if you place some components like that, you will have some
  difficulties when placing 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 of 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
(ALT SPACE) between 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
(ALT SPACE) between the words Place and component.

1.43 Drawing menu/Place reference

When this entry is marked, when components are placed with the mouse, it's reference will also be placed next to 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 held down).
Default references are defined in the libraries and 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 it's 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 components are placed with the mouse, it's 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 held down). 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 components are placed with the mouse, it's pins will be numbered, if this component is defined with pin numbers in the library (always in this case for IC).
If a component is placed on the document and you want to modify this indication, double click on it with the left mouse button and click on the button "Pins numbers" 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 anti-clockwise.
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 (anti-clockwise)

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 mirrored. This operation is done around 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 useful 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 invalidates 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 between (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 (into a cross). To place a line, move the cursor to the start and click the left mouse button, now place 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 these menus: Double line, Bus, Dashed line or Any width.

To modify a line, click on an extremity with the left button, and holding the button down, drag the mouse to another place.

You can also use the keyboard to adjust a line: select the 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
      (ALT SPACE) between the words Place and line.
```

1.51 Drawing menu/Orthogonal mode

```
  When this entry is marked, the lines are always placed horizontally
  or vertically, or at an angle of 45 degrees.
  Select this menu to toggle its state.

Keyboard shortcut: AMIGA-| or |

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

1.52 Drawing menu/Continuous drawing

```
  When this entry is marked, when 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
    (ALT SPACEC) between 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 be drawn at any width. A requester will ask you for the width you want. Enter a value between 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) between the words Dashed and line.

1.57 Drawing menu/Place box

When you select this entry, the cursor changes to 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 held down, place this corner wherever 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) between 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 it's 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 axis' at 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 it's axis', near it's border, and drag the mouse. If you press the CTRL key while you are performing this operation the ellipse will always be drawn as a circle.

To modify the position of an existing ellipse, click in it's center and drag the mouse wherever 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 makes no sense).

Keyboard shortcut: AMIGA-E or E

See also: ELLIPSE.

ARexx call: MENU("Place~ellipse") Warning: solid space (ALT-SPACE) between 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 where you want and click on the left mouse button.

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

You can modify it's 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 provides faster changes.

To edit an existing arc, you can use a double click to select it and use the ALT, CTRL and SHIFT keys in combination with the cursor keys, as above. 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) between Place and arc.

1.60 Drawing menu/Place junction

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) between 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 accept this requester, the text is placed under cursor and you can place it where you want, just use the left mouse button to place 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, LINKVAL, LINKREF.

ARexx call: MENU("Place~text") Warning: solid space
(ALT-SPACE) between 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 place 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) between 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 place 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) between the words.

1.64 Drawing menu/Redraw all

When you select this entry, the complete document is cleared and
redrawn.
It's useful after some selections, block moves, or to display the
grid correctly after some edits...

Keyboard shortcut: AMIGA-Z or Z

ARexx call: MENU("Redraw")

1.65 Menu Edition

These menus are relative to the objects being selected.
You can select some objects with the mouse, in two ways:
- clicking in a zone where there is no object, and with the left mouse button depressed, drag the mouse. All the objects that are present in the rectangle you are drawing will be selected,
- clicking on an object, this one will be selected. However if the menu entry Edition/Multiselection is not marked, only one object at a time can be selected, except if you hold one of the SHIFT keys when you click.

When a block is already selected, you can add objects using one of the SHIFT keys while using the left mouse button.

To select an object that is behind another (to back), select the first object, then click on the second while the ALT key is pressed.

To cancel the selection of an object or of a block, push the CTRL key when you use the mouse (with a SHIFT key if necessary...)

To cancel all the selections, press the ESC key or click twice anywhere in the window.

Copy to clip
Paste from clip
Cut to clip
Delete selection
Clone
Fix on grid
To Front
To Back
Double size
Divide size
New group
Kill group
Save clip
Load clip
Multiselection
Undo

1.66 Edition menu/Copy to clip

This entry handles the copying of the selected objects in memory.
If it succeeds, the selection is canceled.
AmiCAD handles 10 buffers (clips), so you can save 10 different blocks (see CLIPUNIT).

Keyboard shortcut: AMIGA-C (copy) or C

See also: COPY, PASTE.

ARexx call: MENU("Copy")

1.67 Edition menu/Paste from clip

The current clip is copied under the cursor. To place it on the sheet, place the cursor where you want using the mouse and click on the left mouse button.

Keyboard shortcut: AMIGA-V or V

See also: CLIPUNIT, PASTE, COPY.

ARexx call: MENU("Paste")

1.68 Edition menu/Cut to clip

This entry deletes all the selected objects, the block is copied into the current clip.

Warning: This function can modify the numbers of the objects present on the document.

Keyboard shortcut: AMIGA-X or X

See also: CLIPUNIT, PASTE, COPY.

ARexx call: MENU("Cut")

1.69 Edition menu/Delete selection

This entry deletes all the selected objects.

Warning: This function can modify the numbers of the objects present on the document.

Keyboard shortcut: AMIGA-Y, Y or DEL

ARexx call: MENU("Delete")

1.70 Edition menu/Clone

This entry clones the selected objects. You can obtain the same result using first the Copy menu, then the Paste menu.

Keyboard shortcut: AMIGA-TAB or TAB

ARexx call: MENU("Clone")

1.71 Edition menu/Fix on grid

This menu fixes all the selected objects on the grid. This entry is useful to align some objects. The point of the objects that is fixed on the grid is the upper left corner.

Groups are not correctly handled by this menu entry.

Keyboard shortcut: AMIGA-! or !

ARexx call: MENU("Fix")

1.72 Edition menu/To Front

All the selected objects are placed before the others, so when you click on one of them you can reach them easily.

Note: you can reach an object placed behind another one using the ALT key when clicking on them, the first time you'll get the first object, the second one you'll get the second object.

Warning: This function modifies the objects numbering.

Keyboard shortcut: AMIGA-M or M

ARexx call: MENU("To~front") Warning: solid space (ALT-SPACE) between the words To and front.

1.73 Edition menu/To Back

All the selected objects are placed at the end of the list. So when you click on them they are not selected if there are other objects in the same location.

Warning: This function modifies the objects numbering.

Keyboard shortcut: AMIGA-D or D

ARexx call: MENU("To~Back") Warning: solid space (ALT-SPACE) between the words To and Back.

1.74 Edition Menu/Double size

```
  The size of the selected elements is doubled. You can also
  use this function to double the current object under the
  cursor, or while placing an object (from a clip).

  The objects can't go outside the window to permit to the function
  to succeed.

Keyboard shortcut: AMIGA-+ or +

See also: SETSCALE.

ARexx call: MENU("Double~size")     Warning: solid space (ALT-SPACE)
    between the words Double and Size
```

1.75 Edition Menu/Divide size

```
  The size of all the selected objects is halved. The size of these
  objects have to be a multiple of two (scale of 2, 4, 6...)
  This function can also divide the size of the currrent object placed
  under the cursor, or while placing an object (from a clip).

Keyboard shortcut: AMIGA-- or -

See also: SETSCALE.

ARexx call: MENU("Divide~size")     Warning: solid space (ALT-SPACE)
   between the words Divide and size.
```

1.76 Edition Menu/New group

```
  All the selected objects are grouped together. Then you can select
  all of them by clicking on one. The wires connected to a group can
  be used like a component, when this group is moved.
  Grouping the objects of a clip is a good way to create clips that
  look like components.

Keyboard shortcut: AMIGA-{ or {

ARexx call: MENU("New~group")       Warning: solid space (ALT-SPACE)
      between the words New and group.
```

1.77 Edition Menu/Kill group

```
  The selected group is deleted. All the objects that where
  in a group are unselected.

Keyboard shortcut: AMIGA-} or }
```

ARexx call: MENU("Kill~group") Warning: solid space (ALT-SPACE) between the words Kill and group.

1.78 Menu Edition/Save clip

This menu is used to save the selected block in a file, this file is selected with the asl.library requester.

Keyboard shortcut: AMIGA-W or W

See also: SAVECLIP, LOADCLIP functions.

ARexx call: MENU("Save~clip") Warning: solid space (ALT-ESPACE) between the words Save and clip.

1.79 Menu Edition/Load clip

This menu is used to read a block that was previously saved in a file (selected with the asl.library requester). If the operation succeeds the block is displayed under the mouse cursor and is moved with it. Use a left mouse button click to place it, or on the right mouse button to cancel the operation.

The anchor point of a clip is the upper left corner of the box defined by all the objects in this clip. If you create clips with this point on the grid, their placement will be easier later.

Keyboard shortcut: AMIGA-L or L

See also: LOADCLIP, SAVECLIP functions.

ARexx call: MENU("Load~clip") Warning: solid space (ALT-ESPACE) between the words Load and clip.

1.80 Menu Edition/Multiselection

When this entry is marked, you can select all the objects you want, clicking on them without using a SHIFT key.

You have to select this menu to toggle its state.

Keyboard shortcut: AMIGA-*

No ARexx call is possible.

1.81 Menu Edition/Undo

```
  This menu can be used to cancel the last operation that was
  done on the document in memory. But the ARexx scripts or
  ARexx functions will be canceled only if the SAVEALL macro
  is used in them.

Keyboard shortcut: AMIGA-U or U

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

1.82 Menu Macros

```
  These menus are used for macro-commands and
  ARexx scripts.

  Direct mode       Ask for a command, execute it
  Call script       Calls a selected ARexx script.
  ARexx...          20 Programmable ARexx scripts.
```

1.83 Menu Macros/Mode direct

```
  This menu allows the entry of a macro-command, then its execution.
  Note that you can also use the combination of a key ALT and a function key
  to obain 10 preprogrammed macros (see Function Keys).
  It should be noted that these programmed sequences are backed up in the file
  "s:AmiCAD.keys", using the menu Preferences/Keys/Save. Those are then
  reloaded during each launching of the program.

  You can cause the display of a result of a command by beginning the text
  specifying this command with the equal sign (=).

Example:
    =4*95  displays the result of the multiplication (4 times 95)
    =FIRSTSEL  displays the number of the first selected object

Keyboard shortcut: AMIGA-; ou ;

See also: functions EXEC, MACRO, MAP.

ARexx call: MENU("Mode~direct")         Warning: space "solid"
        (ALT-ESPACE) between Mode and direct.
```

1.84 Menu Macros/Call script

```
  This menu allows the entry of a ARexx command, (give the name of the
  script, the extension "AmiCAD" is not obligatory).

Keyboard shortcut:  AMIGA -, or ,
```

See also: function CALL.

ARexx call: MENU("Appel~script") Warning: space "solid"
(ALT-SPACE) between Appel and script.

1.85 Menu Macros/ARexx...

These 20 menuitems are used to define 20 commands able to be launched
by the selection of the corresponding menu.
At the time of the first call of the menu, the user must define
it's command, by giving the name of the script to execute, (without the
path).
The calls which will follow allow the call of the defined script.
The name of the script then appears in the menu. The 20 definitions
can be saved to the file "AmiCAD.prefs", and be reloaded automatically
at the execution of the AmiCAD program.
These menus can be redefined while pressing one of the SHIFT keys
during menu selection.

Note that the editing of a defined script can be made in an automatic
way in a new window by calling the menu while pressing the CTRL key,
(the name must naturally have been defined before).
The program then launches the ARexx script EditScript.AmiCAD, passing to
it the name of this script in argument.
This script, (EditScript), must be adapted to the editor which you use
and to your installation, it's written for my text editor, Amitex.

The command must comprise the name of the script to be executed, (always
without the extension "AmiCAD").
The name can only be a maximum of 11 characters long.
It is not possible to transmit parameters to the script.
The communications with ARexx are made thanks to the port AmiCAD.0,
(or AmiCAD.1, AmiCAD.2... AmiCAD.999 if several tasks are running at the same
time).
Use the command port=ADDRESS() to find the port name, if necessary.

Short cuts keyboard: AMIGA-0 with AMIGA-9

See also: function CALL.

No possible ARexx call for these menus, use function CALL.

1.86 Port ARexx

The program normally opens a port of communication with the ARexx
interpreter during it's launching, this port is called AmiCAD.0, for
the first task being launched. If several windows are opened, the
second port is called AmiCAD.1, then AmiCAD.2 for the following one,
and so (AmiCAD.999 is the limit...)

1.87 Menu Preferences

This series of menus makes it possible to save certain adjustments to the program in the files AmiCAD.prefs and AmiCAD.keys. These files are saved in the current directory.
These files are read automatically at program execution, in order to reload the adjustments desired by the user.
These two files are in ASCII format and can thus be viewed and even modified using a text editor.
You can in fact, create mini-configuration files containing only part of the configuration, like the palette for example, (see the AmiCAD.palette file for an example).

Auto scroll
Backup file
Save icon
Pull lines
Display grid
Horizontal scale
Vertical scale
Sheet size
Screen mode
Palette
Font
Configuration
Function keys

1.88 Menu Preferences/Auto scroll

When this item is enabled, if the cursor arrives at an edge of the current work window, the contents is automatically scrolled if necessary.

There is no keyboard shortcut for this entry.

No possible ARexx call for these menus.

1.89 Menu Preferences/Backup file

When this item is enabled, a backup file with the extension ".bis", is created when the sheet is saved.

There is no keyboard shortcut for this entry.

See also: function SAVECOPY.

No possible ARexx call for this menu.

1.90 Menu Preferences/Save icon

If this item is enabled an icon is created when the sheet is saved.
The name of the program appears in the Default Tool field. The icon
used is the default "Project" icon, stored in directory ENV:Sys.

There is no keyboard shortcut for this entry.

See also: function SAVEICON.

No possible ARexx call for this menu.

1.91 Menu Preferences/Pull lines

If this item is enabled, during the relocation of one or more
components by using the mouse, any wires connected to these
components will be also moved.

Warning: This operation takes place only after having released
these components in their new position.

Keyboard shortcut: AMIGA-F or F

ARexx call: MENU("Pull")

1.92 Menu Preferences/Display grid

If this item is enabled, a grid is displayed on the screen. The step
used is that defined in the menu Drawing/Grid size.
The screen is updated after the call of the menu.

There is no keyboard shortcut for this entry.

No possible ARexx call for this menu.

1.93 Menu Preferences/Full name

If enabled, this menuitem will display the complete name of the
document in the titlebar.
If it is not, the name is displayed without the path, this allows
more space for other indicators in the titlebar, the co-ordinates
of the cursor in particular, when the window is not sufficiently
wide or when the complete name is very long (multiple (sub)-
directories)).

1.94 Menu Preferences/Horizontal scale

The value associated with this menu makes it possible to define the default scale of the components which will be placed on the diagram.

There is no keyboard shortcut for this entry.

See also: functions HSCALE, SETSCALE.

ARexx call: MENU("Horizontal scale")

1.95 Menu Preferences/Vertical scale

The value associated with this menu makes it possible to define the default scale of the components which will be placed on the diagram.

There is no keyboard shortcut for this entry.

See also: functions VSCALE, SETSCALE.

ARexx call: MENU("Vertical scale")

1.96 Menu Preferences/Sheet size

This menuitem makes it possible to define the width and the height of the worksheet. The window is in fact of the SuperBitmap type, which makes it possible to profit from very fast operations, but requires a significant quantity of "CHIP" memory.

There is no keyboard shortcut for this entry.

See also: function DIMSHEET.

ARexx call: MENU("Dimensions")

1.97 Menu Preferences/Screen mode

This menuitem makes it possible to choose the type of screen used to display the windows. An ASL requester is displayed to choose this mode. The low-resolution modes make it possible to work on certain details whereas the high-resolution modes make it possible to have an overall picture of the diagram.

Keyboard shortcut: AMIGA-] or]

See also: function SCREEN.

ARexx call: MENU("Choose~mode") Warning: space "solid" (ALT-SPACE) between Choose and mode.

1.98 Menu Preferences/Palette

This menu makes it possible to choose the colours of the screen used by AmiCAD. Note that these colours are saved in the configuration file. This function requires the reqtools.library.
Ensure that you have a copy of it in your "LIBS:" directory.

Warning: Do not use a too dark a colour for colour 0, black for example, because then the cursor used to place wires would become invisible...

There is no keyboard shortcut for this entry.

ARexx call: MENU("Palette")

1.99 Menu Preferences/Font

This menu makes it possible to choose the font used in the AmiCAD screen. In the absence of the configuration file (see below), it is the font chosen for the default public screen which is used.

Keyboard shortcut: AMIGA-[or [

See also: function SCREEN.

ARexx call: MENU("Choose~font") Warning: space "solid" (ALT-SPACE) between Choose and font.

1.100 Menu Preferences/Configuration

This menu comprises three headings, making it possible to save or read the names of the ARexx scripts as well as, the various modes (run, to save with or without icon, copy diagram...).

LOAD : allows the loading of a configuration file of your choice, saved beforehand using the option SAVE or SAVE AS.

SAVE : allows the saving of the preferences into the AmiCAD.prefs file. This file is loaded at each program execution.
It must be located in the current directory.

SAVE AS : allows the saving of the preferences to a file of your choice (asl.library requester).

No ARexx call for these menus.

1.101 Menu Preferences/Keys

This menu comprises three headings, making it possible to save or read the sequences of programmed keys (function keys, combinations of keys).

LOAD : allows the loading of the configuration file of your choice, saved beforehand using the option SAVE or SAVE AS.

SAVE : allows the saving of the preferences into the AmiCAD.keys file. This file is loaded at each program execution.

SAVE AS : allows the saving of the preferences to a file of your choice (asl.library requester).

See also: function MAP, Mode direct.

No ARexx call for these menus.

1.102 Format of the configuration files

The files AmiCAD.prefs and AmiCAD.keys are ASCII format files, they can thus be viewed, (and also modified), using a simple text editor. Their format follows some elementary rules however.

Each one of these two files starts with a heading which should not be changed, it in particular includes the version number of these files, AmiCAD recognizes only the version in force for a given AmiCAD version. Version 1.2 of AmiCAD uses the headings:
"AmiCADPrefs 1.1" for the AmiCAD.prefs file,
"AmiCADKeys 1.0" for the AmiCAD.keys file.

The AmiCAD.prefs file then includes several paragraphs, these paragraphs can be removed to create a particular configuration, including for example only one palette of colours (AmiCAD.palette file for example).

Each one of these paragraphs begins with texts placed between two hooks ([Menus_ARexx], [Screen], [Palette] and [Mode]), these texts are followed by several lines including each different information. Whole or part of these lines can be removed.

In the same way the AmiCAD.keys file includes two paragraphs [FunctionKeys] and [Macros]. You can add or remove lines in each one of these paragraphs, to suit your configuration.

Warning: The words located on the left of the signs = must be written just as they are (no major/minor changes).

1.103 Writing a line of command(s)

The macro-commands can be carried out in local mode or by the intermediary of an ARexx script. You can refer to the scripts given, for example, in the ARexx directory. These must have the

extension ".AmiCAD" in their name.

Each one of these macro-commands can call one of the ARexx functions, or even several. Some of them require one or more arguments, finally the majority of them return a result. The program also makes it possible to handle variables, (both numerical or character string type). Also you can define your own functions.

The call of a function is done by giving its name, followed by of one or more arguments, surrounded by brackets, possibly separated by commas.

The traditional mathematical operators are of course available:

operator	signs	priority
rise with a power:	^	10
division:	/	9
multiplication:	*	9
modulo:	%	9
addition:	+	8
subtraction:	-	8
AND logic:	&	6
OR exclusive	?	5
OR logic	|	4
assignment	=	2

This last operator (=) can be used to affect variables.

Example:
A=2, assigns value 2 to variable A.

It should be noted that writing A=B=3 is not allowed, it will involve an error message "Impossible Assignment". Rather, write A=3:B=3.

Other operators:

shift on the left:	<<		
shift on the right:	>>		

These two operators have a priority equal to 7.

higher test:	>	higher or equal test:	>=
lower test:	<	lower or equal test:	<=
inequality test:	<>	test equality:	==

The tests return the value 1 if they are true, 0 in the othercases. Their priority is equal to 3.

There is a particular operator which in fact isn't one, it is the sign ':', this allows the separation of two statements, (or more), in order to be able to place several assignments of variables in the same function.
Thus you can place A=2:B=3:C=0 in only one command instead of using three of them. The returned result is then that of the last operation carried out, (0 in this example). This possibility can also be of interest for the function FOR, (see further), in order to carry out multiple initializations at the beginning of a loop.

It should be noted that you can place a comment in a macro comprising a mathematical expression by using the apostrophe seen previously.

Example:
```
DELETE(FIRSTSEL) 'Deletion of the first selected element
```

Lastly, you can place several function calls on the same line, by separating them with a colon.

Example:
```
SETSCALE(FIRSTSEL,200,2000):SETSCALE(NEXTSEL(FIRSTSEL),200,200)
```

1.104 The variables

The variables handled by AmiCAD can be of two types: numeric or character strings. The type is selected at the time of the assignment, it cannot then be changed, unless the variable is reinitialized, (see function INIT).

To assign a value to a variable it is enough to follow its name with the sign = and the value to be assigned to it.

Examples:
```
A = 1
B2 = "text string"
A2 = "First"+B2     gives "First text string"
```

The character strings must be framed by quotation marks if they are constants. Numerical valeurs are limited to integers, (long), use ARexx to handle real numbers.

The names of variables can include from 1 to 21 characters, they must begin with a letter, the following characters are able to be letters (accentuated or not), numbers or the character _.

Examples of valid names:
```
TEST
TYPE_DONNÉE
LINE1
```

Note that the program normally converts the names of functions and variables into capital letters during input, therefore there will not be a difference between two names like variable and VARIABLE, (or even Variable), however in an ARexx script, preferably use capital letters because the strings resulting from the interpreter are not converted, and inevitably the program will then differentiate between names written in lower case and/or upper case.

Warning: The variables are not usable directly with ARexx, they are internal to the program AmiCAD and the ARexx interpreter has it's own variables. It is however possible to simply return a value of variable to the interpreter by giving it's name:

Example:

```
  'NAME_VARIABLE'; v=result
```

The contents of variable NAME_VARIABLE is thus transferred in the variable ARexx v.
It is often advantageous to use internal variables with AmiCAD because their processing is faster, in addition it can avoid the sending of certain messages between the interpreter and the program, which involves an enormous speed profit, (see scripts Zoom and Unzoom).

1.105 Numeric Variables

The numbers handled by AmiCAD are limited to integers. These are signed 32 bit long integers.
The maximum value is 2^31-1 (2147483647) and the minimum value -2^31 (-2147483648).
Use the capacities of the ARexx interpreter to handle numbers in floating point.

1.106 Character string variables

The character strings can include an unspecified number of characters, (theoretically limited only by the memory size of your microcomputer), while starting with null strings which do not include any character.
To define a character string, you must frame it's contents by two quotation marks.
If this string includes itself one or more quotation marks, those must be doubled.

Examples:

```
"This is a character string"
""
"It is a quotation mark "" !"
```

1.107 Structure of ARexx scripts

ARexx scripts are ASCII files, conforming to a format allowing their interpretation by this software. They can include all the functions specific to ARexx and its libraries, (I refer you to their documentation for more precise details), like all the commands supported by the program AmiCAD, (a hundred, plus any functions defined by yourself).

Location
These scripts must be located in the AmiCAD/ARexx directory. It is advised to name them with the AmiCAD extension, but it is not obligatory.

Format
These scripts must always start with a remark included between the characters /* and */, as in language C.

The AmiCAD commands must imperatively appear in upper case. They

are the same ones as those which are used in local mode, however
their analysis by ARexx requires some precautions, thus these commands
must imperatively be framed by apostrophes (') or quotation marks, in
order to differentiate them from the internal functions within ARexx.

Example
The following structure is recommended for these scripts:

```
/* This file gives you an example of structure possible for a
script called by AmiCAD */

options results     /* essential to recover the results of the macros */

signal on error     /* for the interception of errors */
signal on syntax

/* your program must be located in this zone */

exit

/* Processing of errors, interruption of the program */
syntax:
error = RC
'MESSAGE("Syntax error"+CHR(10)+"in line 'SIGL'"+CHR(10)+"'errortext(error)'")'
exit

error:
'MESSAGE("Error in line 'SIGL'")'
exit
```

This example exists already ready in the ARexx directory, under the
name New.AmiCAD.

1.108 List of ARexx functions

```
ABS      computes the absolute value of a number
ARC      places an arc
ARRAYDIM read an element of an array
ASC      returns an ASCII code
ASK      ask for a string from the user
ASKNUM   ask for a number from the user
ASKTEXT  ask for a string from the user
BLINK    draws and clears an object
BOX      draws a rectangle
CALL     calls an ARexx script
CHR      returns the character having the ASCII code specified
CLIPPATH specify the drawer containing clips
CLIPUNIT specify the memory area used for the operations CUT, COPY, PASTE
CLOSE    close the window
COL      number of the column where an object is located
CONVERT  display alternate symbol for component
COORDS   returns the co-ordinates of an object
COPY     copy one or more objects into memory
CURSMODE chose an object placement mode
DATE     returns the actual date
```

```
DEF      defines a new function
DELARRAY removes an array from memory
DELETE   removal of an object
DEVPINS  returns the number of connexions of a component
DIMARRAY set an array
DIMSHEET dimensions the window (SuperBitmap)
DRAW     trace a line
DRAWMODE selection of the type of layout
EDIT     call the edit requester
ELLIPSE  traces an ellipse or a circle
ENDCOL   returns the abciss of an end of line
ENDLINE  returns the number of the line of an end of line
EXEC     interpret a string of characters
FILENAME name of complete file, (including path)
FILEPART name of file
FINDLAB  look for a label
FINDLINE look for a line
FINDOBJ  look for an object
FINDPART look for a component
FINDREF  search for a component by its reference
FINDVAL  search for a component by its value
FOR      processing of a loop
GETARRAY returns the value of an array object
GETCOLOR returns the RGB levels of a given colour
GETDEVS  number of gates in a circuit
GETLABEL read the label number of a line
GETLINE  get a line
GETNET   read the line number associated with a label
GETPART  choose component to use
GETPOINT get a point
GETPOS   returns the placement mode (rotation, symmetry)
GETREF   read the reference of a component
GETVAL   read the value or some type of component
GETZONE  get a box
GROUP    creates a new group
HEIGHT   return the width of a object
HELP     display an AmigaGuide help
HSCALE   return the horizontal scale value of an object
IF       test
INIT     initialization of variable
INPUT    place an input connector
JUNCTION place a connection
LANGUAGE name of the locale
LEN      length of the character string
LIBRARY  return the name of a component library
LIBSPATH specify drawer containing the symbol libraries
LINE     number of the line where to locate a object
LINKLAB  links a label to a line
LINKREF  assignment of a reference with a component
LINKVAL  assignment of a value with a component
LOAD     loading of a diagram
LOADCLIP load a clip
LOADLIB  load a symbol library
LOCK     lock user input
MAP      program a keyboard sequence
MARK     marking of one or more objects
MARKZONE marking of elements in the specified zone
```

```
MENU     executes the function associated to a menu
MESSAGE  display a message
MESURE   return the dimensions of a window
MODIF    test for modification
MOVE     move an object
NBSHEETS number of diagrams present in memory
NEWSHEET
NEXTSEL  number of next element selected
OBJECTS  returns the number of objects
OPEN     open a diagram
OUTPUT   place an output connector
PARTNAME read the name of a component
PASTE    paste the contents of the clipboard
PENWIDTH establish the width of features
PICKOBJ  choose an object using the mouse
PINCOL   returns the column placement of a pin
PINLINE  returns the line placement of a pin
PINNUM   returns a pin number
PRINT    print the diagram
PUTPART  place a component
READCONV read the type of symbol
READDEF  reads the definition of a programmed function
READDEV  read the number of a circuit or a component
READMAP  reads the definition of a programmed key sequence
READTEXT read the text associated with an object
REMLIB   removal of a library
REQFILE  choose a file
REQLIST  choose an item among several
REQSHEET choose a diagram
REQUEST  display a message with a choice of YES/NO
RESET    reinitialization of variables
REXXPORT returns the name of the ARexx port
ROTATE   rotate an object
SAVEIFF  save in IFF format
SAVE     save a diagram
SAVEALL  saves the document in the Undo cache
SAVECLIP saves a clip
SAVECOPY specify the saving of backup files
SAVEICON specify saving with an icon
SCREEN   choose a screen mode
SCRMODE  return the screen mode
SECURITY determine the maximum number of loops before overflow
SELECT   choose an option from several
SETARRAY gives a value to an array object
SETCOLOR set the screen colour
SETLABEL place a label on a wire
SETDEV   choose a circuit or gate of a component
SETFILL  set the filling mode
SETGRID  set the grid step
SETORTHO set mode for drawing lines
SETPINS  display pin numbers
SETREF   set the reference of an object
SETSCALE set the horizontal and vertical scales
SETTEXT  set the text of an object
SETVAL   set the value or the type of an object
SGN      test the sign (+/-) of a number
SHEIGHT  return the height of the screen
```

```
SNAPGRID sets the placement mode
STOBACK  move screen to the back
STOFRONT move screen to the front
STR      conversion of a number into a character string (decimal)
SWIDTH   return the width of the screen
SYMMETRY change the symmetry of an object
TEST     test if an object is selected
TIME     returns the actual hour
TITLE    set the title displayed in a STANDARD window
TOOLBAR  load a toolbar
TRIANGLE places a triangle
TYPE     returns the type of an object
TXHEIGHT returns the height occupied by a text
TXWIDTH  returns the width occupied by a text
UNGROUP  deletes the links of a group
UNLINK   deletes the links of a component
UNLOCK   cancels the locking user input
UNMAP    deletes a programmed key sequence
UNMARK   cancellation of the marking of an object
VAL      conversion of a character string into a number
VERSION  returns the version number of AmiCAD
VSCALE   returns the value of the vertical scale of an object
WHEIGHT  returns the total height of the schematic
WHILE    loop so many times...
WIDTH    returns the width of an object
WINDOW   dimensions the diagram window
WRITE    place text
WTOBACK  moves the active window to the back
WTOFRONT moves the active window to the front
WWIDTH   returns the total width of the schematic
ZOOM     sets the scale of the display
Theme Index
```

1.109 ARexx functional theme classification

```
Processing of Character strings
Place a new object on the diagram
Editing, modification of existing object
Function to implement on a block of objects
Mathematical functions
Interactive functions
Management of windows
Management of preferences
Various functions
```

1.110 ARexx functions for processing of character strings

```
ASC returns an ASCII code
CHR return the character for the specified ASCII code
LEN length of character string
STR conversion of a number to a character string (decimal)
VAL conversion of a character string to a number
```

1.111 ARexx functions to place objects

```
ARC      places an arc
BOX      places a box
CURSMODE chose an object placement mode
DELETE   deletes an object
DRAW     places a line
DRAWMODE select the current line mode
ELLIPSE  place an ellipse
GETDEVS  number of gates in a circuit
GETLINE  get a line
GETPOINT get a point
GETZONE  get a box
GETPART  choose the current component
GROUP    creates a new group
INPUT    place an input connector
JUNCTION place a junction
OUTPUT   place an output connector
PUTPART  place a component
SETFILL  set the filling mode
SETGRID  set the grid step
SETORTHO set mode for drawing lines
SNAPGRID sets the placement mode
TRIANGLE places a triangle
UNGROUP  deletes the links of a group
WRITE    places a text
```

1.112 ARexx functions to manage a block of objects

```
CLIPUNIT specify the memory area used for the operation
COPY     copy of one or more objects into memory
FIRSTSEL return the number of first object selected
LOADCLIP loading of a clip
MARK     marking of one or more objects
MARKZONE marking of elements included in the specified zone
NEXTSEL  number of next element selected
PASTE    joining of content of a buffer report
SAVEALL  saves the document in the Undo cache
SAVECLIP saves a clip
TEST     test if an object is selected
UNMARK   unmark marked object(s)
```

1.113 ARexx functions for managing objects

```
BLINK    draws and clears an object
CLIPPATH specify the drawer containing clips
COL      number of the column where an object is located
CONVERT  display alternate symbol for component
COORDS   returns the co-ordinates of an object
DELETE   removal of an object
DEVPINS  returns the number of connexions of a component
EDIT     call the edit requester
```

```
ENDCOL   returns the abciss of an end of line
ENDLINE  returns the number of the line of an end of line
FINDLAB  look for a label
FINDLINE look for a line
FINDOBJ  look for an object
FINDPART look for a component
FINDREF  search for a component by it's reference
FINDVAL  search of a component by it's value
GETLABEL read the label number of a line
GETNET   read the line number associated with a label
GETPOS   returns the placement mode (rotation, symmetry)
GETREF   read the reference of a component
GETVAL    read the value or type of component
GROUP    creates a new group
HEIGHT   height of an object
HSCALE   return the value of scale horizontal of a object
LIBRARY  return the name of a component library
LIBSPATH specify the drawer containing the symbol libraries
LINE     number of the line where an object is located
LINKLAB  links a label to a line
LINKREF  assignment of a reference to a component
LINKVAL  assignment of a value to a component
LOADLIB  load a symbol library
MOVE     move an object
OBJECTS  returns the number of objects
PARTNAME read the name of a component
PENWIDTH establishes the width of features
PINCOL   returns the column placement of a pin
PINLINE  returns the line placement of a pin
PINNUM   returns a pin number
READCONV read the type of symbol
READDEV  read the number of circuits of a component
READTEXT read a text associated with an object
REMLIB   removal of a library
ROTATE   rotation of an object
SETLABEL place a label on a wire
SETDEV   specify the circuit or a gate of a component
SETFILL  set the filling mode
SETORTHO set mode for drawing lines
SETPINS  display pin numbers
SETREF   fixes the reference of a horizontal object
SETSCALE sets the horizontal and vertical scales
SETTEXT  fixes the text of an object
SETVAL   fixes the value or the type of an object
SYMMETRY symmetry of a STANDARD object
TYPE     returns the type of an object
TXHEIGHT returns the height occupied by a text
TXWIDTH  returns the width occupied by a text
UNGROUP  deletes the links of a group
UNLINK   deletes the links of a component
VSCALE   returns the value of the vertical scale of an object
WIDTH    returns the width of an object
```

1.114 ARexx functions to manage preferences

```
FONTNAME name of character font to use
FONTSIZE size of character font to use
GETCOLOR returns the RGB levels of a given colour
LANGUAGE name of the locale
SAVECOPY choose to save backup copies
SAVEICON choose to create icons for saved files
SCREEN   choose screen mode
SCRMODE  return the screen mode
SETCOLOR set the palette
SETGRID  set the grid size
SHEIGHT  return the height of the screen
SWIDTH   return the width of the screen
TOOLBAR  load a toolbar
```

1.115 ARexx calculation functions

```
ABS      computes the absolute value of a number
FOR      processing of a loop
IF       test
INIT     initialization of variable
RESET    reinitialization of variables
SECURITY number of maximum loops before overflow
SGN      test the sign of a number
STR      conversion of a number into a character string
VAL      conversion of a character string into a number
WHILE    loop so many times...
```

1.116 Interactive ARexx functions

```
ASK      ask for a string from the user
ASKNUM   ask for a number from the user
ASKTEXT  ask for a string from the user
CURSMODE chose an object placement mode
GETLINE  get a line
GETPOINT get a point
GETZONE  get a box
LOCK     lock user input
MESSAGE  display a message
PICKOBJ  choose an object using the mouse
REQFILE  choose a file
REQLIST  choose an item among several
REQUEST  display a message with a choice, YES/NO
SELECT   choose an option among several
UNLOCK   cancels the locking of user input
```

1.117 Various ARexx functions

```
ARRAYDIM read an element of an array
CALL     calls an ARexx script
```

```
DATE     returns the actual date
DEF      defines a new function
DIMARRAY set an array
DELARRAY removes an array from memory
EXEC     interpret a string of characters
FINDLAB  look for a label
FINDLINE look for a line
FINDOBJ look for an objec
FINDPART look for a component
GETARRAY returns the value of an array object
HELP     displays an AmigaGuide help
MAP      programs a keyboard sequence
MENU     executes the function associated to a menu
READDEF  reads the definition of a programmed function
READMAP  reads the definition of a programmed key sequence
SETARRAY gives a value to an array object
TIME     returns the actual time
UNMAP    deletes a programmed key sequence
VERSION  returns the version number of AmiCAD
```

1.118 ARexx functions for window management

```
CLOSE    close the specified window
DIMSHEET dimension the window (SuperBitmap)
FILENAME name of file complete with path
FILEPART name of file
FONTNAME name of character font used
FONTSIZE size of character font used
LOAD     load a diagram
MESURE    return a dimension of window
MODIF    test for modification
NBSHEETS number of diagrams present in memory
NEWSHEET open a new window
OPEN     open a diagram
PRINT    print a diagram
REQSHEET choose a diagram
REXXPORT returns the name of the ARexx port
SAVEIFF  save in IFF format
SAVE     save a diagram
SAVEALL  saves the document in the Undo cache
SETCOLOR specify screen palette
SETGRID  specify grid step
STOBACK  move the screen to the background
STOFRONT move the screen to the foreground
TITLE    specify the title displayed in a window
TOOLBAR  load a toolbar
WHEIGHT  returns the total height of a window
WINDOW   dimensions the diagram window
WTOBACK  moves the active window to the background
WTOFRONT moves the active window to the foreground
WWIDTH   returns the total width of the worksheet
ZOOM     sets the scale of the display
```

1.119 ARexx function DEF

```
  You can define an unspecified number of functions using the internal
  functions like any operator.  One of these functions can even call upon
  another function previously defined.  The arguments can be of an
  unspecified type, it must simply correspond to the types used by the
  functions which will be called or be compatible with the operators used.

  There cannot be more than one declaration in the same line.
  The form of these definitions is as follows:
DEF name_function(argument...) = definition of the operations.
  Key word DEF must imperatively begin the definition, without any
  preliminary space.

  The name of the functions can include/understand from 1 to 13
  alphanumerics, including the accented letters and underlines (_).
  The first character must however always be a letter.
  The internal functions cannot be redefined.  The number of the arguments
  is limited to 15 maximum.
  This number is fixed for a given definition, (you cannot declare a
  function having a variable number of arguments).
  There must be at least one argument in the definition, but this one can
  not be used.
  The arguments must of course have distinct names.

  Examples:
DEF EXPAND(object) = SETSCALE(OBJECT, HSCALE(OBJECT)+1,VSCALE(OBJECT)+1)
  The call of the function will be then made as for all other function:
EXPAND(FIRSTSEL) for example.

  The redefinition of a function having already been defined is possible,
  the new definition then replaces the old one.  This can be useful when
  you try to define a complex function.

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

```
  ARC(x,y,horizontal_radius,vertical_radius,angle_begin,angle_end)

  Draws an arc. The center is specified by the two first arguments
  x and 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 these lines have to be
             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 ARRAYDIM

```
  ARRAYDIM(arrayname)

  This function returns the number of elements of an array, if it exists.
  Else 0 is returned. So you can now if an array has been declared at the
  beginning of a script.

  See also : DIMARRAY, DELARRAY, GETARRAY, SETARRAY
```

1.123 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 between 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
```

1.124 ARexx function ASK

```
This function has been suppressed in version 1.6
Use one of the ASKNUM or ASKTEXT functions instead.

It can be defined using the following definition:
DEF ASK(T)=ASKTEXT(T,"")
```

1.125 ARexx function ASKNUM

```
ASKNUM("title",number)          Version 1.6

Open a requester to get a number.
The second argument is used for the default content of the box.
If the user clicks on Cancel, an empty string is returned,
in the other case, the number in the box is returned.

See also : ASKTEXT.
```

1.126 ARexx function ASKTEXT

```
ASKTEXT("title","text")        Version 1.6

Displays a requester to get a string.
The second parameter is used for the default content of the box.
A null string is returned if the user clicks on Cancel.

See also : ASKNUM.

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

1.127 ARexx function BLINK

```
BLINK(object_number)

The specified object is cleared and drawn three times.
The object number must be set between the values 1 and OBJECTS.
```

1.128 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.
```

1.129 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, (numbers or character strings).

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

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

1.130 ARexx function CHR

```
CHR(code)

Return the character having the specified ASCII code.

Used to obtain a linefeed (CHR(10)) or a special character not present
on the keyboard.  The code can vary from 1 to 255 maximum.

This function can also be used to write special characters, which do
not form part of the character set of the Amiga.  Thus the following
characters are usable:
  - 128:  symbol of amplification (triangle pointed on the right)
  - 129:  open collecting symbol
  - 130:  symbol of OR (greater than or equal)
  - 131:  arrow directed on the right
  - 133:  symbol used for the active inputs (greater than)
  - 134:  symbol of hysteresis
  - 135:  symbol "tri-state" (high impedance)
  - 136:  symbol of an impulse
  - 137:  arrow directed on the left
```

```
    - 138:  symbol describing an analog signal
    - 139:  letter corresponding to the sign "ohm" (for the value of resistance)
    - 140:  letter "alpha" (for potentiometers)
    - 141:  letter TAU (time-constants)

  Test these codes using the macro:
    WRITE(CHR(140), 100, 100)
  You can add one of these characters (or any other to a character string
  by writing the following:
    "Arrow on the right: "+CHR(131)

See also: ASC
```

1.131 ARexx function CLIPPATH

```
  CLIPPATH("path")

  Determine the directory where the clips are located.  This function
  makes it possible to modify the path used for CLIPS.

  The function returns the previous path used.  If the specified path
  is a null string, CLIPPATH(""), there is no modification of the path
  used, only the current path is returned.

Example:
    CLIPPATH("Work:AmiCAD/Clips")
```

1.132 ARexx function CLIPUNIT

```
  CLIPUNIT(unit)

  This function sets the current memory unit used for the Copy/Paste
  functions. The unit number can be from 1 to 10.
  Unit 5 is often used in ARexx scripts, if necessary.

  The value returned by this function is the number of the unit that was
  active BEFORE this call. If you want to know it without changing it,
  give a negative or null argument.

  Warning: if the returned value is null, the function couldn't get
  access to the clip. This can be possible if another task is using
  this clip in the same time (Version 2.05).

Example of script:
    'CLIPUNIT(2)'       select a new unit (number 2)
    clip=RESULT   keeps the old active unit in variable clip
    ...
    'CLIPUNIT('clip')'  select again the old clip unit

See also: COPY, PASTE.
```

1.133 ARexx function CLOSE

```
CLOSE(window)

This function causes the closing of the current window (argument positive
or null) or of all the windows if argument is -1.
There is no returned value.
If a window was modified all the changes will be lost, use the MODIF
function to know if saving is needed before closing the window.

To hide a window, without losing its text in memory, use the
MENU("Hide"), to reduce it to its minimal size, use the
MENU("Iconify").
```

See also: OPEN, LOAD.

1.134 ARexx function COL

```
COL(object_number)

This function makes it possible to know the number of the column where
the specified object is located.
The object number must be set between the values 1 and OBJECTS.
If the argument is null, the left edge of the current selection is
returned (or -1 if there is no selection).
```

See also: ENDCOL, LINE, COORDS.

1.135 ARexx function CONVERT

```
CONVERT(object_number, 0/1/-1)

The object number must be set between the values 1 and OBJECTS.
If it is null, the action will confirm the current mode of placement of
components, (as by using the menu Drawing/Place Component for example).
The returned value will be equal to 0 if the current mode of placement is
the "normal" mode, if the "alternate" mode were valid the returned value
is different from 0.

This function makes it possible to choose the type of symbol for a
component.  According to the value of the second argument, the action
will be the following one:
  - equal to 0:   it is the normal symbol which will be used,
  - equal to 1:   it is the second symbol which will be used,
  - equal to -1:  symbol is changed.

The returned value is then equal to 0 or 1 if the selected object is a
component, if not it is -1.
```

See also: menu Drawing/Alternate symbol, READCONV

1.136 ARexx function COORDS

```
COORDS(object_number)

This function returns the co-ordinates of the object specified in the
form of a character string, separated by commas.
If the object is a line the co-ordinates are the form x0, y0, x1, y1.
For a triangle the form is x0,y0,x1,y1,x2,y2.
For an arc the form is x,y,horizontal_radius,vertical_radius,begin_angle, ↩
   end_angle.
For an ellipse the form is x,y,horizontal_radius,vertical.
They are the form x,y for all other objects.

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

Example of use:
    'COORDS('line')'; coord=result
    /* find the different coordinates */
    PARSE VAR coord x0 ',' y0 ',' x1 ',' y1

See also: COL, LINE, ENDCOL, ENDLINE.
```

1.137 ARexx function COPY

```
COPY(clip)

This function makes it possible to recopy the objects selected in the
specified memory area.  The number of the clip can vary from 1 to 10.
Use the function PASTE to place the objects or choose
the default unit, (CLIPUNIT), then use the menu Edition/Paste from clip.

The returned value is equal to 1 if all occurs normally.
```

1.138 ARexx function CURSMODE

```
CURSMODE(type)          Version 2.00

This function makes it possible to place the cursor in a specified mode:
- if type is equal to 0: clip placement mode,
- if type is equal to 1: component placement mode,
- if type is equal to 2: wires placement mode,
- if type is equal to 3: arcs placement mode,
- if type is equal to 4: text placement mode,
- if type is equal to 7: junctions placement mode,
- if type is equal to 10: ellipse placement mode,
- if type is equal to 22: box placement mode,
- if type is equal to 26: triangle placement mode.
Any other value clears the current mode.

The returned value is the one corresponding to the previous mode (before
applying this function).
```

```
To choose a text or connector placement mode, use one of the
functions WRITE, INPUT or OUTPUT.
```

1.139 ARexx function DATE

```
DATE(day)

Return the current date.  If the argument day is not zero, the day of
the week is included.

Examples:
    DATE(0)     returns 18-Oct-97
    DATE(1)     returns Saturday 18-Oct-97

See also: TIME.
```

1.140 ARexx function DELARRAY

```
DELARRAY(arrayname)

This function frees the memory allocated for an array. If the
array doesn't exists no error occurs, but a 0 is returned.
The number of objects in the array is returned.

See also: ARRAYDIM, DIMARRAY, GETARRAY, SETARRAY
```

1.141 ARexx function DELETE

```
DELETE(object_number)

Remove the object whose number is specified.  Return the number of
remaining objects.

The object number must be set between the values 1 and OBJECTS.
If the argument is null, all the selected objects are deleted,
if the argument is -1, ALL the objects on the document are
deleted (Version 2.01).

Warning:  This function can modify the numbers of the remaining
    objects.
```

1.142 ARexx function DEVPINS

```
DEVPINS(object_number)

This function is used to create a netlist.
The value of the argument must be set between
```

```
1 and OBJECTS.
The specified object must be a  component, the function
then returns the number of connexions of this
component, else an error message is displayed.

See also : PINCOL, PINLINE.
```

1.143 ARexx function DIMARRAY

```
DIMARRAY(arrayname,number_of_objects)

This function allocates some memory to handle an array. Data
placed in arrays can be integers or strings, the first affectation
sets the variable type, like for variables.
The number of objects can be set between 1 and 2^24 (16777216).

See also: ARRAYDIM, GETARRAY, DELARRAY, SETARRAY
```

1.144 ARexx function DIMSHEET

```
DIMSHEET(width,height)

This function allows you to change the dimensions of the current
worksheet. These dimensions are given in pixels.
Use menu Project/Informations to find out the
dimensions of the worksheet.

The limits of size available for the window depend, of
course, on the quantity of CHIP memory installed on your system.
Dimensions of 1100 by 700 are suitable for a system equipped with 2MB CHIP.

Returned value:  1 if the operation succeeded, 0 if it didn't

Warning: No control is made on the values passed in arguments.

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

1.145 ARexx function DRAW

```
DRAW(x0, y0, x1, y1)

Draw a line using the specified coordinates. The line width is dependent
on the current mode (see DRAWMODE).

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

See also : BOX, COORDS.
```

1.146 ARexx function DRAWMODE

```
DRAWMODE(line_type)

Determines which will be the type of line which will be traced (see
DRAW).
If line_type is equal to 0:  they will be dotted lines,
    equal to 1:  they will be "normal" lines (connections),
    equal to 2:  they will be double lines,
    equal to 3:  they will be buses.
If type_line is lower than 0 and higher than -256, lines of a
personalized width will be traced, with a width equal to the absolute
value of that which was specified (v1.1).

The Drawing menu is updated according to the selected type.

This function returns the type of line which was in force BEFORE the
call of the function.
```

1.147 ARexx function EDIT

```
EDIT(object_number)

This function calls a requester making it possible to edit/modify an
element of the diagram.  It has the same effect as a double clicking
on an object using the left button of the mouse.
No value is returned.

The value of the argument must be set between 1 and OBJECTS.

  This function is useful when mapped to a key combination on the keyboard.
Example:
    MAP("alt-e","EDIT(FIRSTSEL)")

You can then select an object and use the combination of ALT-e keys on
the keyboard to call the edit requester on the first selected element.
```

1.148 ARexx function ELLIPSE

```
ELLIPSE(x, y, horizontal_radius, vertical_radius)

Place an ellipse, the center is given by the two first args, x and 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)

If the x and y coordinates are -1, the dimensions of the radius are
used to set the current radius of the ellipses placement mode.
```

```
  Use the COORDS function to get the ellipses datas.

See also: ARC
```

1.149 ARexx function ENDCOL

```
  ENDCOL(object_number)

  This function returns the abciss of the point defining the end
  of a line (dashed line or any width). If the object is another
  type (component, junction or other) the returned value has no
  signification.

  The objet number must be set between 1 and OBJECTS.

  See also : COL, LINE, ENDLINE, COORDS.
```

1.150 Fonction ARexx ENDLINE

```
  ENDLINE(object_number)

  This function returns the number of the line where the specified
  object (a line, any width) is ending.
  If the object is not a line the returned value is no significant.

  The objet number must be set between 1 and OBJECTS.

  See also : COL, LINE, ENDCOL, COORDS.
```

1.151 ARexx function EXEC

```
  EXEC("character string")

  Request the interpretation and execution of the character string passed
  in the argument, as if it were about a line of command.
  The result depends naturally on the contents of the string passed in
  the argument.

Example:
    EXEC(READTEXT(OBJECT))  interprets the line of text associated with the
          object specified
    EXEC(READMAP("CTRL-)")  carries out the sequence associated with the
          combination with keys CTRL -)
    EXEC(READMAP("F5"))     executes the associated sequence for key F5.

  Using a function in the startup file can be usefull:
DEF EXECKEY(X)=EXEC(READMAP(X))
you can there write :
    EXECKEY("F5")
```

```
    or
    EXECKEY("CTRL-)")
```

1.152 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
```

1.153 ARexx function FILEPART

```
  FILEPART("name")

  Renames the current window, preserving the path.
  If a null string ("") is passed in the argument, this function returns
  the name of the current window, without the complete path.

Example:
    Let us suppose that the file running is "RAM:sources/Scheme" function
    FILEPART("") would return Scheme, whereas the function
    FILEPART("New scheme") would rename the file to
    "RAM:sources/New scheme".

See also: FILENAME
```

1.154 ARexx function FINDLAB

```
  FINDLAB(first_object,"Label")

  This function searchs for a wire with the specified label.
  The first argument defines where the search begins. It must be set
  between the values 1 and OBJECTS.

  The returned value is the object number that was found (if any)
  or 0.

  The search is no case sensitive. You can include some AmigaDOS
  jokers (#?) in the label string.

  Note: it's the number of the wire object that is returned,
  not the label one (use GETLABEL to have this one).

  Examples :
FINDLAB(1,"0")  Return the number of the first wire with label 0
FINDLAB(1,"Lab#?")  Retun the number of the first label beginning
    with letters 'Lab'
```

1.155 ARexx function FINDLINE

```
FINDLINE(first_object,x,y)         Version 1.4

This function searchs for a line at the specified coordinates. It
is used to build netlists.
The first argument defines where the search begins. It must be set
between the values 1 and OBJECTS.

The returned value is the object number that was found (if any)
or 0. If the founded line is beginning or ending exactly at the
specified coordinates, the result is in negative form.

Warning: vertical or horizontal lines only are checked.
Other lines are only checked on their extremities.
Only lines with a width of 1 are checked (wires).
To draw frames or anything else than a circuitry, use
the special lines".

Example :
to know if a line is present on coordinates x0,y0
'ABS(FINDLINE(1,'x0','y0'))'; l=result
if l>0 then line_is_present

See also : FINDOBJ
```

1.156 ARexx function FINDOBJ

```
FINDOBJ(first_object,type,x,y)          Version 1.4

This function searchs for an object at the specified coordinates.

The first argument defines where the search begins. It must be set
between the values 1 and OBJECTS.
If this value is set to 1, all the objects on the current document
will be scanned.

The second argument gives the type of object that is searched (as
returned by the TYPE function). If it is 0, the first object
found will be returned.

The coordinates must match an extremity for lines, or the point
where the cursor was at the moment the object was placed. For the
components it is the point where the cursor was when it was placed
(upper left corner). If one of the coordinates is negative, it
is not taken in account, if the two are negatives, the object
will be found elsewhere it is placed.

The returned value is the object number of the object that was found,
else 0.

Examples :
FINDOBJ(1,1,-1,-1)  returns the number of the first component.
FINDOBJ(1,2,100,200) returns the number of a line that have an
```

```
extremity at 100,200, if there is on there.
  FINDOBJ(1,7,50,-1)  renvoie le number of the first junction
place on column 50.

  See also : FINDLINE
```

1.157 ARexx function FINDPART

```
  FINDPART(first_object,"component")

  Begin the search for the component whose name is specified on the
  current document.
  This name corresponds in the name of the symbol, "RESISTANCE" for
  example for a resistance.
  The first argument is used to begin search after an object already
  found.  This argument must take a value ranging between 1 (search
  will be carried out on all the objects of the diagram) and OBJECTS.

  Search is carried out without taking into account the case of letters,
  i.e. upper case and lower case are not differentiated.  In addition,
  you can include "wildcards" (or general characters) in the required
  string, (see AmigaDOS documentation for the various possibilities).

  The returned value is equal to the number of the object which was
  found, or to 0 if no object corresponds.

Examples:
    'FINDPART(1,"RESISTANCE")'; object=result
    'FINDPART(1,"(Res|Cond)#?")'
```

1.158 ARexx function FINDREF

```
  FINDREF(first_object, "reference")

  Begin the search for the component whose reference is specified, R3 for
  example for a resistance or IC9 for an integrated circuit.
  The first argument is used to begin the search after an object already
  found.  This argument must take a value ranging between 1 (search will be
  carried out on all the objects of the diagram) and OBJECTS.

  Search is carried out without taking into account the case of letters,
  i.e. upper case and lower case are not differentiated.  In addition,
  you can include "wildcards" (or general characters) in the required
  string, (see AmigaDOS documentation for the various possibilities).

  The returned value is equal to the number of the object which was
  found, or to 0 if no object corresponds.

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

See also: FINDVAL

1.159 ARexx function FINDVAL

FINDVAL(first_object, "value")

Begin the search for the component whose value or type is specified, 10k for example for a resistance or 7400 for an integrated circuit.
The first argument is used to begin search after an object already found. This argument must take a value ranging between 1 (search will be carried out on all the objects of the diagram) and OBJECTS.

Search is carried out without taking into account the case of letters, i.e. upper case and lower case are not differentiated. In addition, you can include "wildcards" (or general characters) in the required string, (see AmigaDOS documentation for the various possibilities).

The returned value is equal to the number of the object which was found, or to 0 if no object corresponds.

Examples:

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

See also: FINDREF

1.160 ARexx function FIRSTSEL

FIRSTSEL

This function returns the number of the first selected object. It does not require any argument. If no object is marked the returned value is null.

See also: NEXTSEL

1.161 ARexx function FONTNAME

FONTNAME Version 2.05 (argument suppressed)

Return the name of the font used in the current window.

Example:

```
FONTNAME      returns topaz.font (for example)
```

See also: FONTSIZE.

1.162 ARexx function FONTSIZE

```
  FONTSIZE       Version 2.05 (argument suppressed)

  Return the size of the font used in the current window.

Example:
    FONTSIZE      returns 11 (for example)

See also: FONTNAME.
```

1.163 ARexx function FOR

```
  FOR(init,condition_end,action1,...)

  This function makes it possible to define loops.  The first argument
  (init) is carried out only once, when the call of the function has been
  just made.  The second argument defines the condition to end the loop.
  Finally the third argument as well as the following arguments if they
  exist are evaluated with each execution of the loop.

  The use of this function often allows a saving of significant time in
  the execution of a script, the number of messages can decrease to a
  significant degree if it is used advisedly.

Examples:
    FOR (I=0, I<10, I=I+1)
    In this example variable I is initialized with value
    0, as long as this value is lower than 10, one increments this value.
    I, thus will successively take values 1 to 10.

    FOR (I=1, I<=10, I=I+1, DELETE(I))
    This formula includes an additional instruction making it possible to
    remove the first 10 objects.

    N=100:FOR (I=0:J=0, I<=N, J=J+I, I=I+1):J
    This formula makes it possible to calculate the sum of the first 100
    numbers.  The result of the sum is returned (:J at the end).

  Note:  A blocked loop can be stopped by two means:  either by the
  maximum number of loops (defined by function SECURITY,
  or while pressing simultaneously on three keys CTRL, ALT and ESC.

See also: WHILE.
```

1.164 ARexx function GETARRAY

```
  GETARRAY(arrayname,ord)

  This function returns the value of a variable in an
  array. Ord must be included between 0 and the size
  of the array minus 1.
```

```
   See also: ARRAYDIM, DIMARRAY, DELARRAY, SETARRAY
```

1.165 ARexx function GETCOLOR

```
   GETCOLOR(colour)

   This function makes it possible to find the RGB levels of a given colour.
   The number of the colour can vary from 0 to 15.  The levels are returned
   in the form $rrr, $ggg, $bbb in a character string.  This string is the
   same form as that which is in the preference file.

 See also: SETCOLOR.
```

1.166 ARexx function GETDEVS

```
   GETDEVS("circuit_name")         Version 1.4

   This function returns the number of gates in a circuit present in a
   library of symbols. The name can be specified in lower or uppercase.

   If the component doesn't exist or is not in the libraries that are loaded
   in memory, the message "Incorrect argument name" is displayed.

 Examples:
     GETDEVS("LM324")    returns 4 (4 amplifiers)
     GETDEVS("4011")     returns 4 (4 NAND gates)
     GETDEVS("4017")     returns 1 (1 counter)
     GETDEVS("RESISTANCE") returns 0 (no pins number)

 See also: READDEV, SETDEV.
```

1.167 ARexx function GETLABEL

```
   GETLABEL(object_number)

   Return the number of the label object associated to the specified
   object (a line).
   The object number must be the number of a line object.
   If this object is another type, -1 is returned.
   If the label doesn't exist, 0 is returned.

   The object number must be included between 1 and
   OBJECTS.

   Example : READTEXT(GETLABEL(FIRSTSEL))

   See also : SETLABEL, FINDLAB, LINKLAB, UNLINK
```

1.168 ARexx function GETLINE

```
GETLINE("title1","title2")

This function permits to define a line with the mouse. The first
title is displayed in the title bar while waiting for the first
click on the left mouse button by the user. After it's done, the
second title is displayed and a line is drawn when the mouse is
moved. At the second click the coordinates are returned (the
format is the same than with the COORDS function). The function
can be cancelled at any time if the user uses the keyboard or
another mouse button (a null string is returned in this case).

The current preferences are used to draw the line (orthogonal,
fix on grid...)

See also : SETORTHO, SNAPGRID, GETZONE.
```

1.169 ARexx functionGETNET

```
GETNET(object_number)

This function returns the number of the line object linked
to a label. The object number must be the one of a label,
or a value of -1 is returned.

See also : GETLABEL, SETLABEL, LINKLAB, FINDLAB
```

1.170 ARexx function GETPART

```
GETPART("name_component")

This function has been removed in version 2.0. Use function
PUTPART in place.

You can use the next definition to replace it, in the
startup.AmiCAD file, by example, see tooltype STARTUP):
DEF GETPART(C)=PUTPART(C,-1,-1)
```

1.171 ARexx function GETPOINT

```
GETPOINT("title")       Version 1.6

This function displays the specified message in the title bar
and returns the coordinates of the point where the user clicks
with the left mouse button.
These coordinates are returned under the form column*32768+line.
To get the coordinates, you can use something like that in an
ARexx script:
```

```
'GETPOINT("Click on a point")'
p=result
col=bitand(p,'FFFF0000'x)
col=col/32768
line=bitand(p,'FFFF'x)

If the user clicks on the right mouse button or on the keyboard
the returned value is -1 (operation cancelled).

The next macros can be defined to recover the coordinates of the
point :
DEF POINTLINE(P)=P&0xFFFF
DEF COLONN(P)=P>>15
```

1.172 ARexx function GETPOS

```
GETPOS(object_number)

Returns a value defining the placement mode of a component,
a text or a connector. If the object was placed without any
rotation, this value is equal to 1, else it can be 2 to 4.
If the specified object had a symmetry, the returned value
is negative.

If the object is another type the returned value is always 0.

See also : ROTATE, SYMMETRY.
```

1.173 ARexx function GETREF

```
GETREF(object_number)

Returns the reference's object number that's associated with the
specified component.
The number of the object passed in the argument must naturally be a
component.
If the object is of another type the function returns -1.
If this reference does not exist the function returns 0.

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

Example : READTEXT(GETREF(FIRSTSEL))

See also: SETREF, LINKREF, UNLINK, GETVAL
```

1.174 ARexx function GETVAL

```
GETVAL(object_number)

Returns the value's object number that's associated with the specified
```

```
  component.
  The number of the object passed in argument must naturally be a
  component.
  If the object is of another type the function returns -1.
  If this reference does not exist the function returns 0.

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

Example : READTEXT(GETVAL(FIRSTSEL))

See also: SETVAL, LINKVAL, UNLINK, GETREF
```

1.175 ARexx function GETZONE

```
  GETZONE("title")          Version 2.05 (Second argument removed)

  This fonction permits to define a box using the mouse. The title
  is displayed in the title bar during the operation.
  To define the zone, the user must click on the left mouse button,
  keeps it while movin the mouse, and leaves it when the mouse is at
  the second corner.
  Any action on the keyboard or on another button cancel the
  function (a null string is then returned).
  If the function succeeds, a string defining the coordinates is
  returned (like by the COORDS function.

  See also : GETLINE.
```

1.176 ARexx function GROUP

```
  GROUP(object_number1,object_number2...)

  Create a new group with all the specified objects.
  Returns the number of the new group.

  GROUP(object_number)            Only one argument
  Returns the group number of the object (0 if not included in a group).

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

See also: UNGROUP.
```

1.177 ARexx function HEIGHT

```
  HEIGHT (object_number)

  Returns the height of the specified object, in pixels.
  The object number must be set between the values 1 and OBJECTS.
  If it's null, the height of the selected block is returned (-1
  if nothing is selected).
```

```
See also: WIDTH
```

1.178 ARexx function HELP

```
  HELP("node")

  This function allows the call of AmigaGuide, as by the menu
  Project/Help. The argument must be the name of a node, (node),
  of the AmiCAD.guide file. You can specify the name of any function or
  menu.

Examples:
    HELP("Copy~to~clip")
    HELP("DRAW")
```

1.179 ARexx function HSCALE

```
  HSCALE(object_number)

  This function returns the value of the horizontal scale of the specified
  object.
  The object number must be set between the values 1 and OBJECTS.
  If the argument is null, the current horizontal scale is returned (v2.07).

See also: VSCALE, SETSCALE.
```

1.180 ARexx function IF

```
  IF(x, a1, a2)

  If x is different from zero, returns a1 if not returns a2.  Only a1 OR a2
  will be evaluated, according to the result of x.
  Note that the arguments a1 and a2 can be of any type, they can also call
  upon other functions, including other IF functions.

Examples:
    IF(A>B,A,B)                 returns the maximum value of A and B
    DEF MIN(A,B)=IF(A<B,A,B)    definition function MIN
    IF(GETPART(ASK("Component?")),MENU("Placer~composant"),0)
```

1.181 ARexx function INIT

```
  INIT(variable,...)

  This function is identical to function RESET, however the type is
  re-initialized, i.e. that variable will be able to then take any
```

```
authorized type, (numerical or character string).
The number of arguments is unspecified.

Example :
A=1         Variable A is numeric type
...
INIT(A)          A is initialized (the type is loosed)
...
A="String chars"    Variable A is now a string

See also: RESET
```

1.182 ARexx function INPUT

```
INPUT("name", x, y)

Place a connector with the specified name, at the co-ordinates x, y.
The arrow of the connector is directed towards the left.
If the function succeeds, it returns the number of the object, if not 0.

The current vertical scale and the horizontal scale are taken into
account to determine the dimensions of this element, (see SETSCALE,
ROTATE, SYMMETRY).

See also: OUTPUT
```

1.183 ARexx function JUNCTION

```
JUNCTION(x,y)

Place a junction at the specified coordinates. The current 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.
```

1.184 ARexx function LANGUAGE

```
LANGUAGE      Version 2.07

Retuns the name of the "locale" used by the system.
For the french the string "français.language" is returned.
This function permits to localize the ARexx scripts.
```

1.185 ARexx function LEN

```
    LEN("string")

    Return the length of the character string passed in the argument.
```

1.186 ARexx function LIBRARY

```
LIBRARY (object_number)

    The name of the library where is located the specified component
    is returned, if any, in another case, a null string is returned.
    The name is complete (with path).

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

    Voir aussi : PARTNAME, PUTPART.
```

1.187 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 find the location of the libraries.

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

1.188 ARexx function LINE

```
    LINE(object_number)

    This function returns the number of the line where the specified object
    is located.
    The object number must be set between the values 1 and OBJECTS.
    If the argument is null, the top edge of the current selection is
    returned (or -1 if there is no selection).

  See also: COL, COORDS.
```

1.189 ARexx function LINKLAB

```
LINKLAB(object1,object2)

This function makes it possible to link a label to a line. An
object must be a text, and it must not be connected to another
wire or another component (value, reference). The other argument
must be a line number.

The arguments must be include between 1 et OBJECTS, they
can be placed in any order.

Example : LINKLAB(FIRSTSEL,NEXTSEL(FIRSTSEL))

See also: FINDLAB, GETLABEL, SETLABEL, UNLINK
```

1.190 ARexx function LINKREF

```
LINKREF(object1,object2)

This function makes it possible to assign an reference to a component.
The reference object must be of the text type and does not have to be
already related to another component.

The object numbers must be set between the values 1 and OBJECTS,
they can be placed in any order.

Example :
  LINKREF(FIRSTSEL,NEXTSEL(FIRSTSEL))

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

1.191 ARexx function LINKVAL

```
LINKVAL(object1,object2)

This function links two objects, one must be a component and
the other a text. The component must be without a value before
this operation, the text is associated to it as its value.

The object numbers must be set between the values 1 and OBJECTS,
they can be placed in any order.

Example :
  LINKVAL(FIRSTSEL,NEXTSEL(FIRSTSEL))

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

1.192 ARexx function LOAD

```
LOAD("name_file")

Load the file diagram specified into the current window.  The window
loses its contents, even if it had been modified.  Use command MODIF
to know if the diagram was modified since the last save.  The window
takes the name of the file which was loaded.
Return 0 if it succeeded, if not an error code.

See also: OPEN, SAVE
```

1.193 ARexx function LOADCLIP

```
LOADCLIP(unit_clip,"file")

Load the file specified into the specified unit.  If the name of the
file is not specified, the search is carried out in the current path,
then in the path specified by the tooltype CLIPS.

Warning:  The clip is not placed under the cursor or on the diagram,
use the menu Edition/Paste from clip or the function PASTE
for that.

See also: CLIPUNIT, SAVECLIP.
```

1.194 ARexx function LOADKEYS

```
LOADKEYS("macro_file")

This function has been suppressed inversion 2.05
```

1.195 ARexx function LOADLIB

```
LOADLIB("name_library")

This function makes it possible to load a library of symbols into
memory, it returns 0 if the library was loaded.
If the name of the library does not include a directory or "device"
(DF0:, Work:, etc...), a search will be carried out in the current path
then in the path specified by the tooltype LIBS.

Example : LOADLIB("Symbols CMOS")

See also: REMLIB, LIBSPATH.
```

1.196 ARexx function LOADPREF

```
LOADPREF("nom_fichier")

This function has been suppressed inversion 2.05
```

1.197 ARexx function LOCK

```
LOCK      Version 2.05 (argument removed)

Lock the current window. All during the execution of
the script the user cannot have access to the document any more:  the
menus are not accessible and any operations made using the keyboard or
the mouse are not taken into account.
If a locking is carried out on an already blocked window, a counter is
incremented, it is necessary then that the window is freed as many times
as it was blocked before it is actually unlocked.

Returned value: code equal to or higher than zero if successful operation,
-1 if impossible to lock, (lack of memory?).

Foot-note: The windows are automatically freed when the execution of a
script finishes.  If there is a blocking you can also put an end to
this situation by clicking twice on the program's AppIcon located on the
Workbench screen.  This function cannot be called via macro since there
would be risk of totally blocking the keyboard input/output.

See also: UNLOCK.
```

1.198 ARexx function MACRO

```
MACRO(x)

This function has been removed in version 2.04. It can be replaced
using something like EXEC(READMAP(key)).
```

1.199 ARexx function MAP

```
MAP("combination keys", "sequence")

This function makes it possible to program a macro-command, which will be
executed when a specific key combination is used.
Keys ALT, SHIFT and CTRL can be used as qualifiers, (it is necessary to
use at least one of them, except for function keys).
Only single character keys are definable (for the moment): you cannot
define, using this function, either some special keys or the arrows.

The sequence can include any other command, it must be included  between
brackets, (character string).  The definition can start with the sign =,
the result of the macro is then automatically displayed, if there is one.
```

```
  Note that these definitions are saved in the AmiCAD.keys configuration
  file when the Preferences are saved, and are thus reloaded at the
  execution of the program.

Examples:
    MAP("shift-ctrl-a","SAVE(""Prog:Projects/Schemes/New scheme"")")
    MAP("CTRL-)","CALL(""Swap"")")
    MAP("F10","WTOBACK")

See also:
    READMAP, UNMAP, definition of Function keys.
```

1.200 ARexx function MARK

```
  MARK(object_number,...)

  Allows the selection of one or more objects.
  The object numbers must be set between the values 1 and OBJECTS.
  This function returns the number of objects which were actually selected,
  (the objects which were already selected are not counted).

See also: UNMARK, MARKZONE.
```

1.201 ARexx function MARKZONE

```
  MARKZONE(x0, x1, y0, y1)

  Allows the selection of objects included in the rectangle defined by the
  co-ordinates x0, y0 and x1, y1.  The elements must be entirely in the
  zone to be selected.
  This function does not return any particular value.

  To select the whole diagram, use the following command:
    MARKZONE(0,0,WWIDTH(-1)-1,WHEIGHT(-1)-1)

See also: MARK, UNMARK.
```

1.202 ARexx function MENU

```
  MENU("menu title")

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

```
Examples:
    MENU ("Quit")   Close all the documents and exit the program.
    MENU ("Copy")   Copy the selected objects in the current clip.

  Return: the menu name if the menu entry was found, else an error
  message is displayed (Incorrect argument value).

  Note: The menu title has to match the localized strings or the
  english built-in string (better: the script will work for any
  country).

Example: MENU("Copier") in French becomes MENU("Copy") in English.
```

1.203 ARexx function MESSAGE

```
  MESSAGE("string")

  Display a requester containing the specified message.  This text can
  contain some linefeeds.

Example:
    MESSAGE("This is one"+CHR(10)+"message.")
```

1.204 ARexx function MESURE

```
  MESURE(window_dimension)

  Return one of dimensions of the current window.  The value which is
  returned depends on the argument:
    equal 0:  co-ordinate of the window's left edge
    equal 1:  co-ordinate of the window's top edge
    equal 2:  width of the window (in pixels)
    equal 3:  height of the window (in pixels)
    equal 4:  maximum width able to be taken by the window
    equal 5:  maximum height able to be taken by the window
    equal 6:  width of screen (ditto SWIDTH)
    equal 7:  height of screen (ditto SHEIGHT

  If the window is hidden the returned value is always null, except for
  dimensions of the screen, null only if the screen is closed, (when all
  the windows are hidden).
  If the window is iconified the value determining its dimension is
  returned in negative form, (for argument values 0 to 3 inclusive).

See also: WINDOW
```

1.205 ARexx function MODIF

```
  MODIF   Version 2.05 (argument removed)

  Allows you to know if the current window has been modified
  since it was last saved.

  The returned value is null if the diagram was not modified since the
  last save, if it is was modified then it will normally equal 1.

Example:
    'MODIF(-1)'
    if result~=0 then 'MENU("Save")'

    Another method (faster):
    'IF(MODIF(-1),MENU("Save"),0)'

    This small script can be used for autosaving (see script SauverAuto.AmiCAD)) ↩
       .
```

1.206 ARexx function MOVE

```
  MOVE(object_number, dx, dy)

  This function makes it possible to move an object the number of pixels
  specified by the values dx and dy.

  The object number must be set between the values 1 and OBJECTS.
  If it's null, all the selected objects will be moved.

  No value is returned by this function.
```

1.207 ARexx function NBSHEET

```
  NBSHEET(x)

  This function has been removed in version 2.05 (see NBSHEETS).
```

1.208 ARexx function NBSHEETS

```
  NBSHEETS("variable_name")                  Version 2.05

  This function is used to know how many windows are opened, and
  which ARexx port they are using. It can't be used in local mode.
  The variable name is the name of an ARexx variable that will
  hold the names of the ports:
variable_name.0 contains the name of the first window port
variable_name.1 contains the name of the second window port
Thee number of windows is contained in variable result.

  Example :
```

```
options results
'NBSHEETS("NAMES")'
n=result
do i=0 to n-1
    address value NAMES.i
    'MESSAGE("Document "+FILENAME("")+": port 'NAMES.i'")'
end
```

1.209 ARexx function NEW

```
NEW("title")

This function has been removed, use NEWSHEET function in place.
```

1.210 ARexx function NEWSHEET

```
NEWSHEET         Version 2.05

This function opens a new window. If it succeeds the result is
positive or nul. If the returned value is negative the function
failed. The returned value is the number of the new task, the
ARexx port of this new task is AmiCAD.x (where x is the result).

The size of the new window is defined by the preferences (document
size), they can be changed using DIMSHEET function.

Use the NewWindow script if you want to open a new window, the
returned value, in some cases, can be no significant if the
window couln't be opened, this script handles these problems.

Example:
call 'NewWindow.AmiCAD' [filename]
code = result
if code < 0 then exit
address value "AmiCAD."||code
These four lines can open a new window that will take the name
passed as an argument (filename). The returned code is the one
returned by function NEWSHEET.
```

1.211 ARexx function NEXTSEL

```
NEXTSEL(object_number)

Return the number of the next selected object, according to that which
is specified in argument.  This function allows you to traverse all the
selected elements when used in association with FIRSTSEL.
```

1.212 ARexx function OBJECTS

```
OBJECTS        Version 2.05 (argument removed)

This function returns the number of objects that are present on the
current window.
```

1.213 ARexx function OPEN

```
OPEN("name_file")

This function makes it possible to load a file in a new window,
the difference to function LOAD being it also allows
the loading of several files.
It is possible to specify generic characters (#?[]) in the name of the
file, as envisaged under DOS.
All the diagrams corresponding to this pattern will be loaded.
A new window is opened for each diagram found, corresponding to the
request.
The function returns the number of windows that were opened (v2.07).

Examples :
    OPEN("Travail:AmiCAD/Schemes/Projet_TV/#?")  asks for the loading of
        all the files of the directory Projet_TV
    OPEN("#?.sch")  request to load all the files having the sch extension,
        located in the current directory

See also: CLOSE, LOAD.
```

1.214 ARexx function OUTPUT

```
OUTPUT(nom_connecteur, x, y)

This function makes it possible to place an input connector at the
specified co-ordinates x and y.
It returns the number of the object which was placed if it succeeds,
if not it returns 0.

If the two coordinates are null, the specified string is placed
under the cursor, and the current mode becomes output connector
placement.

The current vertical and horizontal scale are taken into account to
determine the dimensions of this element, (see SETSCALE, ROTATE,
SYMMETRY).

See also: INPUT
```

1.215 ARexx function PARTNAME

```
PARTNAME(object_number)

This function makes it possible to know the name of a component.
It returns a null string if the selected object is not a component.

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

See also: PUTPART, GETPART
```

1.216 ARexx function PASTE

```
PASTE(clip, x, y)

Place the contents of the specified buffer (clip) at the co-ordinates
given by the arguments x and y.

No particular value is returned.

See also: COPY, CLIPUNIT.
```

1.217 ARexx function PENWIDTH

```
PENWIDTH(object_number,width)

Allows you to select the width of the feature defining the layout of the
specified object.  The value of the second parameter must lie between
1 and 255, if not the function will not have any effect.

Returned value: Width of the feature used BEFORE the execution of the
function.

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

1.218 ARexx function PICKOBJ

```
PICKOBJ("message")

Display the message specified in the titlebar of the window, it then
awaits a click of the left mouse button in the window.
Return the number of the object on which the click took place, (0 if
there isn't one).
The user can scroll the text using the sliders and associated buttons.
You can also cancel the operation by pressing on the right mouse button
or on a key of the keyboard, the returned value is then -1.
```

1.219 ARexx function PINLINE

```
PINLINE(object_number,pin)    Version 1.4

This function returns the line where is located the specified
pin. The object must be a component.

The pin value is not the pin number, when it exists, but the
order in which the pins are present in a Spice netlist.

See also : DEVPINS, PINCOL, PINNUM.
```

1.220 ARexx function PINNUM

```
PINNUM(object_number,pin)      Version 1.4

This function returns the number of the specified pin.
The object must be a component.

The pin value is not the pin number, when it exists, but the
order in which the pins are present in a Spice netlist.

See also : DEVPINS, PINCOL, PINLINE.
```

1.221 ARexx function PRINT

```
PRINT(ratio, rotation)

Print the diagram, taking account of the
arguments: ratio (factor of enlarging, must be equal to or higher than
1) and rotation (if equal to 0, the diagram is printed as it is displayed
on the screen, if equal to 1, the diagram is printed with a rotation of
90 degrees).

Return 0 if all occurs well, an error code in the contrary case.
```

1.222 ARexx function PUTPART

```
PUTPART("name_component", x, y)

Place the component specified at the co-ordinates given by the following
arguments.
Warning: the reference and the value are not placed, use functions
SETREF and SETVAL or LINKREF and LINKVAL for that.

The currant vertical and horizontal scale are taken into account to
determine the dimensions of this element (see SETSCALE, ROTATE,
SYMMETRY).

This function returns the code of the object which was placed if it
succeeds, if not it returns 0.
```

```
  Version 1.6:
  If the coordinnates x and y are equal to -1, the specified component
  becomes the current component, when using the Drawing/place component.
  This function replaces the old GETPART function.

See also: GETPART, PARTNAME
```

1.223 ARexx function READCONV

```
  READCONV(object_number)

  This function makes it possible to know what the type of symbol used to
  display a component is.
  It returns -1 if the selected object is not a component, 0 if this
  component is displayed normally or 1 if it is the alternate symbol which
  is used.

  The object number must be set between the values 1 and OBJECTS.
  If it's null, the current mode is returned.

See also: CONVERT, menu Drawing/Alternate symbol
```

1.224 ARexx function READDEF

```
  READDEF("function")

  This function makes it possible to know the definition associated with
  a function.  The name of the function passed in the argument must appear
  in capital letters, between brackets.  This function must of course
  have been defined beforehand using function DEF, if not a null string is
  returned.
  You cannot read the internal definitions, like ABS, ASK, etc...

Example:
    READDEF("CIRCLE")   returns the definition associated with function CIRCLE,
      if it exists.

See also: DEF
```

1.225 ARexx function READDEV

```
  READDEV(object_number)

  This function makes it possible to know which is the gate or the
  circuit used by a component.
  It returns -1 if the selected object is not a component, 0 if this
  component includes only one circuit or the number of the selected
  circuit, (then it will be equal to or higher than 1).
```

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

  See also: SETDEV, GETDEVS.
```

1.226 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-$\mathrm{\mu}$")
      EXEC(READMAP("F1"))       Executes the associate command of key F1

  See also: MAP, UNMAP
```

1.227 ARexx function READTEXT

```
    READTEXT(object_number)

    Return the text associated with an object, this object can be an
    element of text, or a connector (input or output).
    If this text does not exist or if the object is of a different type
    the function returns a null string.

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

  See also: SETTEXT
```

1.228 ARexx function REMLIB

```
    REMLIB("library_name")

    Remove the specified library from memory.

    Warning:  no element of this library must be present on
    an opened document, or this operation will fail.

    This operation can make it possible to gain memory, by removing for
    example, a library which is not used any more.

    You can also use the button "Remove" in the "Loading component"
    requester.

  Example : REMLIB("Symboles TTL")
```

See also: LOADLIB

1.229 ARexx function REQFILE

```
REQFILE("title_requester","path","file")

Open a file requester with the specified title and in the specified
path.  This path must correspond to that of a directory or a volume.

Return the full name of the selected file or a null string if the user
clicks on the Cancel button.

Example:
REQFILE("Choose a file","Work:AmiCAD","")
```

1.230 ARexx function REQLIST

```
REQLIST (title, entry1, ...)          Version 2.04

This function opens a requester with the specified title, the
entries are displayed in a listview, the number of the selected
entry is returned if the user clicks on one of them (1 for the
first item). If the user clicks on Cancel or closes the window,
-1 is returned.

If the requester couldn't be opened, 0 is returned.
```

1.231 ARexx function REQSHEET

```
REQSHEET("title")              Version 2.05 (modified)

This function makes it possible to choose, using a requester, a
diagram among those which are loaded in memory.
The title is displayed in the first line.
The name of the ARexx port of the selected diagram is returned.
If the user presses on the right mouse button or if a problem occurs, a
null string is returned.
Operation is similar to that obtained after a double click on the
right mouse button, however with this function you can choose
the title displayed in the heading.
```

1.232 ARexx function REQUEST

```
REQUEST("title")

Display a requester with the specified text, (maximum of thirteen lines,
separated by linefeeds), and two buttons YES and NO.
```

If the user clicks on the button YES or uses the Enter key, this function returns 1.
If the user clicks on the button NO or uses the ESC key, the returned value is 0.
In the event of a problem, (lack of memory for example), the returned value is negative.

1.233 ARexx function RESET

RESET(variable,...)

This function is identical to function INIT, however the type is not re-initialized.
If variable is of the numerical type it takes value 0,
if it is a character string it becomes a null string ("").
The number of arguments is unspecified.

Example:
RESET(A,B,C)

See also: INIT

1.234 ARexx function REXXPORT

REXXPORT("nom_fichier") Version 2.05

This function returns the name of the ARexx port associated to a document, if it's present in memory. A null string is returned if the document doesn't exists.
The argument doesn't needs to be the full path name (optional). If it's a null string, the name of the current window is returned.

See also : NBSHEETS

1.235 ARexx function ROTATE

ROTATE(object_number, rotations)

Fact of turning the specified object the number of quarters of specified turn.
If the object number is null, it is the current mode of placement which is affected: the placements carried out then will take account of this new position, that it is by macro (PUTPART for example) or a menu.
This function returns 0 if the operation failed, (for lack of space for example), if not it returns 1.

See also: SYMMETRY, SETSCALE

1.236 ARexx function SAVE

```
SAVE("file_name")

Save the document under the specified name.  If the file exists already
no warning is given.
Return 1 if no problem occurs, if not 0.
An info file comprising an icon is created if the menu
Preferences/Save icon is marked, a file comprising
the ".bis" extension is created if the menu Preferences/Backup file
is marked.

See also: SAVECOPY, SAVEICON.
```

1.237 ARexx function SAVEALL

```
SAVEALL     Version 2.05 (argument removed)

This function is used to save all the documents that are present in
memory in the Undo cache.  So you can later cancel the changes made by
an ARexx script.
The function must be called before you make any change to the document.
The user can cancel the operations done by the script, (or macro), using
the Undo menu.

Examples:
  SAVEALL(-1)     saves the current document
  SAVEALL(0)      saves the first document
  SAVEALL(1)      saves the second document
```

1.238 ARexx function SAVECLIP

```
SAVECLIP(unit,"filename")

This function is used to save the specified clip in a file.
Naturelly the clip must have been initialized using the COPY
function, for example.

See also: LOADCLIP, CLIPUNIT.
```

1.239 ARexx function SAVECOPY

```
SAVECOPY(1/0/-1)

This function makes it possible to set, like the menu
Preferences/Backup file, whether a copy of the
diagram will be created at the time of saving of the diagram.
This function returns 1 if the menu was marked BEFORE the execution of
the function, or 0 if it wasn't it.
```

If the argument is -1 the menu state is toggled.

To read the state of the menu without modifying it, use the command SAVECOPY(-2).

See also: SAVEICON.

1.240 ARexx function SAVEICON

SAVEICON(1/0/-1)

This function makes it possible to set, like the menu Preferences/Save icon, whether an icon will be created at the time of saving of the diagram.
This function returns 1 if the menu was marked BEFORE the execution of the function, or 0 if it wasn't.
If the argument is -1 the menu state is toggled.

To read the state of the menu without modifying it, use the command SAVEICON(-2).

See also: SAVECOPY.

1.241 ARexx function SAVEIFF

SAVEIFF("file name",ratio)

The current document is saved using the IFF format (2 colors).
A 0 is returned if it worked.
Version 2.01: the ratio is given in percentage (equal to 100 to have a normal size).

Example :
SAVEIFF("Work:Images/Schéma_amplificateur", 200)

See also: Project menu/Save IFF

1.242 ARexx function SAVEKEYS

SAVEKEYS("filename")

This function has been removed from version 2.05.

1.243 ARexx function SAVEPREF

SAVEPREF("filename")

This function has been removed from version 2.05.

1.244 ARexx function SCREEN

```
SCREEN (mode, width, height,"font",font_size)

This function makes it possible to choose the resolution of the screen
of AmiCAD without having to use the ASL requester, as with the menu
"Preferences/Screen mode".  Moreover you can specify which will be the
font used for the menus, requesters, etc...
The name of the font must be complete, including the extension ".font".
However it will be necessary for you to know the values associated with
the various types of screen, a simple means to find out is to display
them with function SCRMODE, by typing the command
=SCRMODE into the menu Macros/Direct requester.

  Thus the following values are possible:
  - 561188 for a screen in super interlaced resolution (SUPER72)
  - 561152 for a screen in SUPER72 high resolution.
  - 233509 for a screen MultiScan interlaced Productivity
  - 167936 for a screen STAKE high resolution
  - 135168 for a screen STAKE low resolution

The function returns the code in force BEFORE its execution.

The program does not correctly manage, for the moment, dimensions
higher than the visible portion of the screen.

The screen always includes 16 colours.

Examples:
  SCREEN(561188,800,600, "courier.font",15) "normal" screen (Super 72)
  SCREEN(135168,320,200, "topaz.font",8)    low resolution screen (for " ↩
     closeup")

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

1.245 ARexx function SCRMODE

```
SCRMODE

This function returns the value associated with the mode with current
screen.
It does not need any argument.

See also: SCREEN
```

1.246 ARexx function SECURITY

```
SECURITY(number_of_loops)

This function determines the maximum number of loops able to be carried
out by one of the functions FOR or WHILE.
This makes it possible to exit endless loops rather simply.
```

The default value is equal to 500. You can enter a value going from 1 until 2^31-1 (2147483647).
The function returns the value in progress before its execution.

You can give this function an argument of great value without fearing blocking, it is now possible to stop a loop while pressing simultaneously on the keys CTRL, ALT and ESC.

If the value passed in argument is null only the current value is returned, without modification.

1.247 ARexx function SELECT

```
SELECT("text")
```

This function makes it possible to open a requester comprising a variable number of buttons, each one using a text chosen by the user.
If the user clicks on one of these buttons, the function returns the row of the button, (while starting with value 1 for the first, 2 for the second, etc...).

The format of the text passed in argument must be the following:
- a first line, displayed in title,
- a second line, corresponding to the text of the first button,
- a third line, corresponding to the text of the second button,
- and so on, as many lines than of buttons...

Each line is separated from following by a linefeed (CHR(10)).

If the returned value is negative or null, the user pressed on the right mouse button or the Esc key, or the requester could not be opened.

Example:

```
SELECT("Number of circuits?"+CHR(10)+"10 circuits"+CHR(10)+"20 circuits"+CHR ↩
    (10)+"Undetermined")
```

1.248 ARexx function SELFILE

```
SELFILE("file_name")
```

This function has been removed from version 2.05.

1.249 ARexx function SELSHEET

```
SELSHEET(window)
```

This function has been removed from version 2.05.

1.250 ARexx function SETARRAY

```
SETARRAY(arrayname,ord,value)

This function sets a value to an object of an array. Ord
must be included between 0 and the array size minus 1.
If ord is lower than 0, ALL the objects of the array are
sets to the specified value.
The value must always be of the same type (integer or string)
for a given array.

Example:
'DIMARRAY(N,100)'
'FOR(I=0,I<100,SETARRAY(N,I,I*2),I=I+1)'
'GETARRAY(N,50)'            return 100
'SETARRAY(N,-1,0)'
'GETARRAY(N,50)'            return 0

See also: ARRAYDIM, DIMARRAY, DELARRAY, GETARRAY
```

1.251 ARexx function SETCOLOR

```
SETCOLOR(colour,red_level,green_level,blue_level)

This function makes it possible to choose the colour of the AmiCAD
screen.
The number corresponding to this colour must lie between 0
and 15 inclusive, (the screen has 16 colours).
The level of the colours are long words, 32 bits. To specify white,
pass three arguments equal to 0xFFFFFFFF, to bring a level of colour
to an intermediate level, pass an argument of value 0x7FFFFFFF.

No value is returned by this function.
```

1.252 ARexx function SETDEV

```
SETDEV(object_number, circuit_number)

This function makes it possible to choose the number of the circuit or
gate for a component that is comprised of several of them.
It is useful for the circuits comprised of multiple gates like TTL 7400,
for example.

The object number must be set between the values 1 and OBJECTS.
If it's null, the current placement mode is affected (the next component
that will be placed will have this number).

The function can return three possible results:
   - 0  if the specified object is not a component,
   - 1  if the operation succeeded,
   - -1 if the number of the circuit is incorrect.
```

See also: READDEV, GETDEVS.

1.253 ARexx function SETFILL

SETFILL(0/1/-1)

Define the filling mode when drawing boxes, triangles or ellipses (v2.0).

If argument is 1, the objects will be filled, if it's 0 they will
not. If it's -1 the mode is unchanged.
The function returns the value that was used before execution of the command.

1.254 ARexx function SETGRID

SETGRID(grid_step)

Define the size of the grid used to place the components on the diagram.
Note that this function does not update the grid on the screen, which
can pose problems, use the menu "Drawing/Redraw all" if you wish that
it be updated, by using for example the following command:
  SETGRID(15):MENU("Redraw all")

Returned value: The step value used before the execution of the command.

Use the command SETGRID(0) to know the current grid step without
modifying it.

See also : SNAPGRID.

1.255 ARexx function SETLABEL

SETLABEL(object_number,"Label")

The specified label (a string char) is affected to the specified
object. If this object already had a label, it's replaced.
Naturely, this object must be a line.

The object number of the added or modified object is returned
if the operation succeeded (0 else). This number is the same
that the returned value of the GETLABEL function (Version 2.08).

Example : SETLABEL(FIRSTSEL,"Label1")

See also : LINKLAB, UNLINK, GETLABEL

1.256 ARexx function SETORTHO

```
SETORTHO(action)

Define if the lines will be drawn using 45\textdegree{} angles or not, when ↩
   they
are placed by the user.

See also menu Drawing/Orthogonal mode.

If action is equal to 1, the menu will be marked,
if action is equal to 0, the menu is unmarked,
if action is equal to -1, the menu state is toggled.
For any other value nothing happens.

Returned value: the value used before the execution of the command.
```

1.257 ARexx function SETPINS

```
SETPINS(object_number,ON/OFF)

This function makes it possible to determine whether the pin numbers of
a component will be displayed or not.
If the second argument is higher than 0, the display is confirmed, if
it is equal to 0, the display is removed, if it is less than 0 nothing
is changed.

The returned value is equal to 1 if the display was validated BEFORE
the application of the function, 0 in the contrary case.

If the specified object is not a component, the returned value is always
equal to -1.

  Example:
SETPINS(FIRSTSEL, -1) returns the state of the 1st object selected
```

1.258 ARexx function SETREF

```
SETREF(object_number,"reference")

The specified value, (in the form of a character string), is allocated to
the specified object.  If the specified object already had a reference,
it is replaced.

Example:
  SETREF(FIRSTSEL,"R2")

See also: LINKREF, SETVAL, LINKVAL
```

1.259 ARexx function SETSCALE

SETSCALE(object_number, horizontal_scale, vertical_scale)

This function is used to modify the scale of an object.

The object number must be set between the values 1 and OBJECTS.
If it's set to 0, the function sets the current scales, used to place new objects, like 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, it's value will not be changed.

This function returns 1 if it succeeded, 0 if it didn't.

If a script has to change one of the current placement scales, it is good to set this scales to their first value. You can do this using the next example script :

```
/* We want to use horizontal scale X and vertical scale Y */
'_EH_=HSCALE(0):_EV_=VSCALE(0):SETSCALE(0,X,Y)'
...
'SETSCALE(0,_EH_,_EV_)'
```

See also : HSCALE, VSCALE

1.260 ARexx function SETTEXT

SETTEXT(object_number,"string")

This function makes it possible to set the text associated with an object.
This object must be of the type text, value or reference of a component, or a connector.

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

The returned value is equal to 1 if the function succeeded, if not it is null.

See also: READTEXT

1.261 ARexx function SETVAL

SETVAL(object_number,"value")

The specified value, (in the form of a character string), is allocated to the specified object. If the specified object had already a value, it is replaced.

Example:

```
  SETVAL(FIRSTSEL,"100k")

See also: LINKVAL, SETREF, LINKREF
```

1.262 ARexx function SGN

```
SGN(number)

Return 1 if the argument is positive, -1 if it is negative, zero if it
is null.
```

1.263 ARexx function SHEIGHT

```
SHEIGHT

This function returns the height of the current screen.
It does not require any argument.

See also: SCREEN, SWIDTH, SCRMODE
```

1.264 Fonction ARexx SNAPGRID

```
SNAPGRID(action)

This function sets the placement mode of the objects by the user.

If action is equal to 1, the menu will be marked,
if action is equal to 0, the menu will be unmarked,
if action is -1, the menu state is toggled.
For any other value nothing happens.

The menu state before execution of the function is returned.

See also: SETGRID.
```

1.265 ARexx function STOBACK

```
STOBACK

This function moves the AmiCAD screen to the background.
It does not require any argument.

See also: STOFRONT WTOBACK WTOFRONT
```

1.266 ARexx function STOFRONT

```
STOFRONT

This function moves the AmiCAD screen to the foreground.
It does not require any argument.

See also: STOBACK WTOBACK WTOFRONT
```

1.267 ARexx function STR

```
STR(number)

Return the character string corresponding to a number.
The base used for conversion is base 10 (decimal).
```

1.268 ARexx function SWIDTH

```
SWIDTH

This function returns the width of the current screen.
It does not require any argument.

See also: SCREEN, SHEIGHT, SCRMODE
```

1.269 ARexx function SYMMETRY

```
SYMMETRY(object_number, position)

This function makes it possible to mirror an object about it's vertical
axis, (or its horizontal axis, if it's rotated one or three quarters of
turn).

The object number must be set between the values 1 and OBJECTS.
If this number is null, the function modifies the way of placing
components, (menu "Drawing/Place component").

The position argument can take three values:
   - equal to 1:   symmetry is carried out,
   - equal to 0:   no symmetry,
   - equal to -1:  position reversal.

The returned value is equal to 1 if there was a modification of the
position of the object, 0 in the contrary case.

See also: ROTATE, SETSCALE
```

1.270 ARexx function TEST

```
TEST(object_number)

Allows you to find out if an object is selected or not.
The object number must be set between the values 1 and OBJECTS.

The function returns 1 if the object is selected, 0 in the contrary case.
```

1.271 ARexx function TIME

```
TIME(seconds)

Return the current hour, if the argument is non null, the seconds are
included.

Examples:
  TIME(0)                   returns 23:24
  TIME(1)                   returns 23:24:27
  WRITE(TIME(1),0,0)        writes the current time "under" the cursor

See also: DATE
```

1.272 ARexx function TITLE

```
TITLE("title")

Specify the title of the current window.  If the argument is a null
string, the title reverts to it's normal state, i.e. the name of the file
reappears there.
This function makes it possible to announce a long operation by warning
the user, without blocking the window or the program.

Example of script:
  'TITLE("Processing...")'
  'TITLE("")'                 gives the normal title
```

1.273 ARexx function TOOLBAR

```
TOOLBAR("filename")

The specified toolbar is loaded. The file format must be IFF.
Its icon must have some tooltypes :
-COLUMNS: number of icons on a line,
-LINES: number of icons lines,
-COMMAND(x,y): command associated to the icon located
at the specified coordinates (a command for an icon).
-XBAR: window left edge (optional),
-YBAR: window top edge (optional).
```

This function uses the iff.library to load the images.

To create a new toolbar, create an image with a
program like PPaint, DPaint or any other one. Save this image
as a brush, keeping one pixel on the borders (all the icons
must have the same dimensions). Then you can edit the
image tooltypes, adding the COLUMNS and LINES tooltypes
as a minimum. The commands can then be edited using
AmiCAD : load the toolbar, select it using the left mouse
button, then click on any icon with the right mouse button.
A requester is opened, choose a button to select the action
you want to do (edit, copy, paste, shift or save commands)
Don't remember to save the result when all is
working like you want.

Warning: if you are using a paint program to edit the image,
be carefull : some of these programs are clearing all the
tooltypes when saving your work, if so, work on a copy of
the image.

This function returns 0 if loading the image was fine (version
2.05).

1.274 ARexx function TRIANGLE

TRIANGLE(x0,y0,x1,y1,x2,y2)

This function draws a triangle using the specified coordinates.
The line width uses the current drawing mode (see DRAWMODE)
and the filling mode (see SETFILL).

The number of the object that was created is returned (0 if error).

Use function COORDS to know the coordinates of an
existing triangle.

1.275 ARexx function TXHEIGHT

TXHEIGHT("text")

Return the height of the specified text, in pixels. This function
takes into account the current scale, as well as the mode of placement,
(possible rotation).
In all the cases it is the occupied vertical space which is returned.

See also: TXWIDTH

1.276 ARexx function TXWIDTH

```
TXWIDTH("text")

Return the width of the specified text, in pixels.  This function
takes into account the current scale, as well as the mode of placement,
(possible rotation).
In all the cases it is the occupied horizontal space which is returned.

This function is useful to center a text in a rectangle:
  options results
  text="Test text"
  xx=100;y=100
  'TXWIDTH("'text'")'
  l=result
  'TXHEIGHT("'text'")'
  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("'text'",'xx','y+h')'

See also: TXHEIGHT
```

1.277 ARexx function TYPE

```
TYPE(object_number)

This function makes it possible to find out the nature of an object.

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

The returned value depends on the type of the character:
  - equal to 1:   it is a component
  - equal to 2:   it is a normal wire connection
  - equal to 3:   it is an arc of a circle
  - equal to 4:   it is a text
  - equal to 5:   it is a reference of a component
  - equal to 6:   it is a value of a component (or its type)
  - equal to 7:   it is a connection
  - equal to 8:   it is a feature in dotted lines
  - equal to 9:   it is a bus
  - equal to 10:  it is an ellipse
  - equal to 11:  it is an input connector
  - equal to 12:  it is an output connector
  - equal to 15:  it is a double line
  - equal to 21:  it is a personalized line (unspecified width)
  - equal to 22:  it is a box
  - equal to 24:  it is a filled box
  - equal to 26:  it is a triangle
  - equal to 27:  it is a filled triangle.

You can thus define functions which recognize the objects:
  DEF COMPONENT(O) = IF (TYPE(O) == 1, 1, 0)
  DEF WIRE(O) = IF (TYPE(O) == 2, 1, 0)
  DEF CONNECTOR(O) = IF ((TYPE(O)==11) | (TYPE(O)==12), 1,0)
```

```
  DEF LINED(O) = IF ((TYPE(O)==2) | (TYPE(O)==8) | (TYPE(O)==9) | (TYPE(O) ↩
     ==15) |  (TYPE(O)==21), 1, 0)
```

Each one of these functions returns 1 if the character is of the type tested, if not they return 0.

1.278 Arexx function UNGROUP

UNGROUP(group) Version 1.4

The group whose number is specified as the argument is dissolved.

Returns the number of objects that were in the group.

See also: GROUP.

1.279 ARexx function UNLINK

UNLINK(object_number)

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

This function makes it possible to break the links existing between a component and it's reference, or it's value. The object number can be the component, the reference or the value. The function returns the number of links that were broken.
If the specified object is not a component returned value is equal to -1.

Version 2.08: this function also permits to break the link between a label and a line. The object number can be any of the two objects : the label or the line.

Examples:

```
    UNLINK(FIRSTSEL)
    UNLINK(PICKOBJ("Click on a component"))
```

See also: LINKVAL, LINKREF

1.280 ARexx function UNLOCK

UNLOCK Version 2.05 (argument removed)

Cancel the locking of the current window, initiated by function LOCK.

1.281 ARexx function UNMAP

```
  UNMAP("key combination")

  This function deletes a programmed macro associated with a key
  combination.
  The ALT, SHIFT and CTRL can be used to define the combination, (one of
  them must be used, except for the function keys).

Examples:
    UNMAP("shift-ctrl-a")
    UNMAP("ctrl-shift-a")
    UNMAP("CTRL-)")
    UNMAP("ALT-$\mathrm{\mu}$")
    UNMAP("F2")

See also:
    MAP, READMAP
```

1.282 ARexx function UNMARK

```
  UNMARK(object_number,...)

  Cancel the marking of the specified objects.

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

  Return the number of selections having been cancelled.

  See also: MARK, MARKZONE
```

1.283 ARexx function VAL

```
  VAL("string")

  Return the numerical value corresponding to the character string passed
  in the argument.
  Only the characters corresponding to integers are taken into account.
  The number must appear in decimal form, except if it is preceded by the
  prefix $, which announces that it is in hexadecimal form.

  Examples:
    VAL("14")   returns 14
    VAL("14.3") returns 14
    VAL("$5F")  returns 95 (the $ specifies a hexadecimal string)

  See also: STR
```

1.284 ARexx function VERSION

```
VERSION(arg)

This function returns the current version number of the program,
the argument can be any value (version 2.03).
This function can be used to test if a program can execute a script or
not.

Example of script:
  'VERSION(0)'
  if result < 2.00 then do
'MESSAGE("This program version"+CHR(10)+"can''t do this job!")'
exit
  end
  ...
```

1.285 ARexx function VSCALE

```
VSCALE(object_number)

This function returns the vertical scale value of the specified object.

The object number must be set between the values 1 and OBJECTS.
If the argument is null, the current vertical scale is returned (v2.07).

See also: HSCALE, SETSCALE.
```

1.286 ARexx function WHEIGHT

```
WHEIGHT(window_index)            Version 2.05 (argument removed)

The returned value corresponds to the width of the document, in pixels.

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

1.287 ARexx function WHILE

```
WHILE(end,action1,...)

Is identical in function to FOR, it just misses this
functions first argument.
The initialization of variables or the units tested
will thus have had to be carried out before.

See also: SECURITY
```

1.288 ARexx function WIDTH

```
WIDTH(object_number)

Returns the width of the specified object, in pixels.

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

See also: HEIGHT
```

1.289 ARexx function WINDOW

```
WINDOW(x, y, width, height)

This function allows you to redimension the current window, or to move
it.
The first two arguments correspond to the co-ordinates of the top left
corner of the window on the screen.  The two following arguments
determine its dimensions.

Warning: This function functions only for "normal" windows.
For iconified windows, only co-ordinate x can be modified, (to move the
windows).
The hidden windows cannot, naturally, be modified but you can specify
command WINDOW(-1, -1, -1, -1) to allow their reopening.

If you do not want to modify one or more these parameters and you do not
know their value, give them a negative value (-1).
If the arguments are too large or aberrant values, the program will try
to cure it as well as possible.

The returned value is equal to the number of values having been taken
into account.

Examples:
  WINDOW(0,0,-1,-1)       moves the window in top on the left of the
        screen without modifying its dimensions.
  WINDOW(-1, -1, 200, 50) exchange dimensions of the window

See also: MEASURE, tooltype WINDOW
```

1.290 ARexx function WRITE

```
WRITE("string", x, y)

Place a text at the specified coordinates. If the coordinates are null,
the string is placed under the cursor and the user can click anywhere
to place it.

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.

1.291 ARexx function WTOBACK

WTOBACK Version 2.05 (argument removed)

Send the current window to the background.

See also: WTOFRONT, STOBACK, STOFRONT

1.292 ARexx function WTOFRONT

WTOFRONT(window) Version 2.05 (argument removed)

Send the current window to the foreground.

See also: WTOBACK

1.293 ARexx function WWIDTH

WWIDTH Version 2.05 (argument removed)

Return the working width of the curent window.
The returned value corresponds to the height of the document, in pixels.

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

1.294 ARexx function ZOOM

ZOOM(ratio)

This command can modify the scale of the display. The ratio can be
between 25 (scale is then equal to 25%) and 1000 (maximum zoom).
If the argument is outside these values, the function has no effect.

The returned value is the scale that was used before the function was
executed.

1.295 Use of the keyboard

```
HELP

Launches the program AmigaGuide, this one loads the help file
AmiCAD.guide.
This guide must be in the directory where the AmiCAD programis located,
or at the place specified by the tooltype HELPFILE.

Edit commands

DEL:  Erasure of the selected objects

CURSOR keys

ARROW (LEFT/RIGHT/UP/DOWN):
The selected elements are moved, pixel by pixel or 10 pixels if the
SHIFT key is also pressed.

ALT ARROW:
Edit the rays of the selected arc, pixel by pixel or 10 pixels if the
SHIFT key is also pressed.

CONTROL ARROW:
Edit the angles of the selected arc, degree by degree or by variation
of 10 degrees if the SHIFT key is also pressed.

CTRL:
Allows deselection of elements if used at the same time as the left
button of the mouse during the selection.

TAB:  copy (clone) selected elements

ALT HELP: brief display of the role of keys ALT with the function keys.
ALT F1/F10: programming of macro-command, then replaying.
SHIFT ALT F1/F10: reprogramming of the macro-commands.

Utilisation of the FUNCTION keys

F3       :  open a new window, choosing a file to load
F4       :  open a new window
F9       :  move the screen behind
F10      :  move the window behind
```

1.296 Utilisation des touches de fonction

```
The ten function keys can be associated with sequences of commands
including calls to functions which you will have defined.
These programmed sequences are saved in the file "s:AmiCAD.keys",
using the menu Preferences/Keys/Save file.
These will then be reloaded during each execution of the program.

The call of a combination is done while pressing on ALT and the key of
the desired function.  The redefinition of this key can be done while
pressing on keys SHIFT and ALT before pressing on the function key to
redefine.
```

```
Example:
    - the programmed sequence SETGRID(5) is on the key F1,
  a) press ALT, SHIFT and finally F1,
  b) a requester is opened, showing the macro there:
SETGRID(5)
  c) confirm, the definition is saved in memory,
  d) press now the keys ALT and F1, the macro is carried out.

  To redefine it later on, start again at a).
```

1.297 bgui.library

```
  The author of the BGUI library is Jan van den Baard.  This library is
  available in the public domain, (Aminet Sites for example).

BGUI release 1.1
(c) Copyright 1993-1994 Jaba Development
(c) Copyright 1993-1994 Jan van den Baard
Written with compiler DICE v3.0 by

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

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

1.298 Online Help

```
  An online help can be obtained constantly, using the AmiCAD.guide file.
  This file must be located in the same directory as the program or at a
  place and a name specified in the tooltype HELPFILE
  of the AmiCAD program icon, (this information is taken into account
  during the execution of the program).

  To launch the assistance you can press on the HELP key, immediately or
  during the selection of a menu, which enables you to have direct
  assistance concerning this menu.  You can also use the menu
  Project/Help and give the name of a node, (node), of the help
  file.  You can thus find help very quickly for any function, by giving
  it's name.

  Note that you can also get help for an error caused by a function.
  Press on the HELP key when the requester announcing the error is
  displayed.
```

1.299 Useful macros

The following definitions can be integrated into the AmiCAD.keys file, either using a text editor, or by defining them under AmiCAD (see function MAP) then by choosing the menu Preferences/Keys/Save file.

Macro to allow choosing a component by typing only the first letters of it's name (even only the first):

```
IF((PP=ASK('Component?'))<>'', GETPART(PP):MENU('Place'), 0)
```

Macro to allow a call to a function associated with another macro, for all the selected objects, (here it is MACRO(5) which is called, that is to say the macro defined on key F5):

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

Selection of all the elements on the document:

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

Loading of a library of components by requester, without using the BGUI requester:

```
LOADLIB(ASK("Library to load?"))
```

or better:

```
IF((LIB=ASK("Library to load?"))<>"", LOADLIB(LIB), 0)
```

Addition of the sign "ohm" to the value of a resistance, (the VALUE of resistance in question must only be selected):

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

Loading of a clip, move this clip under the cursor:

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

To bind a text to a component, (as a value or a reference):

```
LINKVAL(FIRSTSEL, PICKOBJ("Click on the value of this component"))
LINKREF(FIRSTSEL, PICKOBJ("Click on the reference of this component"))
```

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

This program required much work by me so I would kindly request you to be lenient for the always possible errors occurring during it's use. I would be grateful to you for agreeing to inform me if you note one or more anomalies.

The use of the program under a CyberGraphics screen can sometimes give odd results, (the function of COMPLEMENT drawing mode does not function or very badly, for area fills, whereas the same program functions perfectly under AGA modes).

It can happen that a System Error occurs during the execution of an ARexx script, if a command causes an error. This problem is normally avoided by following the structure given in the example script New.AmiCAD for ARexx scripts.

The AmiCAD screen is a public screen, thus you can open one, (or even several), other application on it's screen. However, always close these other windows BEFORE leaving AmiCAD, if not there can be some problems...

1.301 History

```
Version 2.08   March 25, 2001
    New Label object (connected to wires).
    New functions GETLABEL, LINKLAB, SETLABEL, FINDLAB, GETNET.
    Functions SETREF, SETVAL modified (returned values).
    New functions ARRAYDIM, DIMARRAY, DELARRAY, GETARRAY and SETARRAY (to handle
    data arrays).
    Mousemove events handler modified (uses now the interval defined
    in the Preferences of the system for a double click before moving
    anything, to avoid troubles while clicking fastly on objects).

Version 2.07a   January 1, 2001
    Bug correction modulo operator (%)

Version 2.07   Novembre 12, 2000
  New function LANGUAGE (to localize scripts).
  Bug correction loading a clip (many hits Enforcer if component not found).
  Modification of functions HSCALE and VSCALE to get the
  current scales.
  Checking boxes (clicks) more accurate.
  Some enhancements to this guide (history, some new functions added...)
  If the program was launched from shell with an argument, if this
  argument was a file that was not found, the program didn't display
  an error message: bug corrected.
  New swedish catalog by Henrik Isaksson: henisak@algonet.se
  Bug correction saving function keys (a _ char was joined to the string
  defining the key name).

Version 2.06   Novembre 1, 2000
  Bug correction to the key handler. Gtlayout.library version number changed (45).
  New button to get a functions list requester on macro-commands requester.
  New "Application" button on edition requesters to extend or not the
  modification on other objects.
  New finnish catalog by Joni Halme: halmej@saunalahti.fi

Version 2.05   Avril 14, 2000
  Bug correction when pulling lines (a wire was sometimes moved twice).
  Bug correction for printing (could lead to a crash if the last line was
  filled)
  Printing by bands (request less chip memory, can print larger
  documents, even with a high resolution, like when using laser printer).
  Multitask (each window is handled by a separate process),
  modification of the ARexx interface : some functions has been
  removed (NEW, SELFILE, LOADKEYS, LOADPREF,
  MACRO, SAVEKEYS SAVEPREF, SELSHEET, NBSHEET),
  some functions have been added: (NEWSHEET, NBSHEETS, REXXPORT)
  or modified (CLOSE, MODIF, LOCK, UNLOCK, SAVEALL, OBJECTS, FONTNAME,
  FONTSIZE, WWIDTH, WHEIGHT, GETZONE, REQSHEET, WTOFRONT, WTOBACK).
  Some options availables with Shell has been suppressed :
  MACRO, SHEET_WIDTH, SHEET_HEIGHT, some tooltypes has also been
  removed : WINDOW, MACRO, GRIDSIZE.  These parameters can be
  fixed using the new preferences menu.

Version 2.04   (November 28, 1999)
  Bug correction of value/reference placement to the left of a component.
```

```
  Modification of the arcs handler (slower but cleaner).
  Lines width requester modified (was buggy if a special line was edited)
  Extension is now possible for boxes, arcs, triangles, ellipses.
  Bug correction of AmiCAD2META (caused a crash of the Amiga when
  being runned in previous version), thanks to Arno Richter).
  New component "Bornier" in Symboles_électriques library.
  New german catalogs for symbol libraries AmiCAD, "divers" and "logiques"
  (Thanks to Gu0cky). New english catalog for "logiques" symbols (myself).
  File requester handler enhanced (AmiCAD windows refresh is possible
  while a requester is opened).
  Modication of function keys handler: we can now program these
  keys using any qualifier (ALT, SHIFT and (or) CTRL) or not.
  A new menu and requester has been added to program these keys.
  ARexx MACRO macro suppressed.
  Nouvelle REQLIST function.

Version 2.03   (September 26, 1999)
  New requesters for editing objects.
  Bug correction of MAP function, new MapKey script.

Version 2.02   (August/September 1999)
  Replacement of the bgui.library by the gtlayout.library (code is
    now shortest and faster).

Version 2.01   (Mai 1999)
  Som bugs correction of display when using "Auto-scroll" mode.
  Modification of the function COORDS to get data from
    angles and ellipses (radius, angles).
  New toolbar handler (TOOLBAR).
  Modification of the MENU function for using menu headers
    given in english, whith any locale being used.
  Modification of functions DELETE and SAVEIFF.
  New function LIBRARY.
  New Edition/Scale menu to change the scale of objects.

Version 2.00   (March 1999)
  New functions CURSMODE, SETORTHO, SNAPGRID, GETLINE, GETZONE.
  Modification of the objects scale handler (now given in percentage:
    100 gives a normal scale).
  New files format (use the ARexx script ImportFile_1.0 to load your old
  files).
  New zoom (from 25 to 1000%, see also ARexx function ZOOM).
  Bug correction of loosed allocated memory when loading a clip.
  Bug correction of functions SELSHEET and SELFILE that caused
    the selected windows to be locked.
  Middle mouse button now used to select the current document.
  Modification of function REQFILE.

Version 1.6 (not distributed)
  Type of connector now displayed (Input or Output) when editing.
  ENTER key used to edit the first selected object, or to place an objet.
  Modification of functions COL, LINE, HEIGHT and WIDTH to know the
    placement and dimensions of the current selected objects.
  Bug correction of clips rotation display.
  Modification of connectors handler (alternate symbol, symmetry).
  Modification of the UNLINK function (unlinking value or reference
    only is now possible).
```

```
  Bug correction for printing square junctions (version 1.5 crashed...)
  Bug correction for handling boxes symmetry.
  Bug correction when copying with no current selection (the current clipboard was ↩
      flushed).
  New GETPOINT, ASKTEXT and ASKNUM functions.
  Functions ASK and GETPART suppressed (replaced by ASKTEXT and PUTPART).

Version 1.5 November 2, 1998
  New menus defined by the user.
  Adaptation of the utility AmiCAD2META to the format of libraries,
  bug reported by Henk Jonas.
  New requester to permit to stop printing before it's finished.
  Saving faster (thanks to Sébastien VEYRIN-FORRER).
  Saving the menu "Preferences/Full name" state now works (report from David Beryl ↩
     ).
  Correction bug function PENWIDTH.
  New square junctions (using the menu "Drawing/Alternate symbol").
  Problem opening screen requester in PAL mode solved... (?)

Version 1.4
  New ARexx functions GETDEVS, GROUP, UNGROUP.

Version 1.3 June 1, 1998
  Correction of the buggy function "To restore it" (finally...?)
  Corrections bugs processing boxes (impression and clips)...
  Correction bug function SETPINS. Addition function BOX.
  Modification of functions WRITE, INPUT and OUTPUT.

Version 1.2 April 12, 1998
  Corrections bugs:  function DATES, use sometimes failing functions
  user (DEF), functions SETDEV, SETGRID, REMLIB...
  Addition functions SAVEALL and GETCOLOR.
  Improvement of the management of small Symmetry.  Addition of the
  processing of the operations of rotation, symmetry, enlarging and
  reduction on the clip in progress.
  Passage of the screen from 8 to 16 colours:  the elements are now
  drawn different colours, according to their type.
  Addition of the right-angled object (not of dotted lines for the
  moment).
  Addition of a menu in the preferences to keep or not the complete
  name of the file in the bar of title.  Modification of the
  indications carried in the bar of title (suggestion of Sebastien
  VEYRIN-FORRER).
  Improvement of script Roasts (creation of a grid in the rectangle
  defined by the user).
  Addition of the Italian catalogue (by Massimo Basso), new version
  of the Spanish catalogue (Benjamin Morente).
  Localisation of the symbols library.

Version 1.1 March 8, 1998
  Replacement of the file of Configuration.AmiCAD configuration by the
  file AmiCAD.prefs (more need to use ConfigFile.library, however the
  new files of configuration are not compatible with the old ones).
  Improvement of the script of installation.
  Writing of the German catalogue (by Henk Joans).
  Modification opening screen (screen of size equal to that of Workbench
  by defect).
```

```
  Correction of some small bugs (width of the buttons in the requests,
  width of the iconified windows, management macro ARexx SYMMETRY) as
  well as few others more significant (macro LOADCLIP).
  Addition tooltypes SHEET_WIDTH and SHEET_HEIGHT.
  Some optimizations of the code (shorter program).
  Addition edition width milks component, texts, ellipses, arcs, etc.
  Modification management of the arrows cursor for displacement of the
  objects and edition arcs and ellipses.
  Modification of the management of the groups.
  Addition of the function ARexx SETPINS.
  Suppression of the use of ConfigFile.library, replaced by internal,
  shorter routines.
  Correction bug colours screen AGA.
  Suppression of the ' requesters ', replaced by windows...

Version 1.00  18 janvier 1998
  First version, written for system 3.0. 1st diffusion on Aminet.
```

1.302 Help-me!!

```
    This program can be very enhanced. But I have not got all the docs that
    I need and not much 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 following items:
    - description of the EPSF format (for saving under this format, to load
      the sheets with a program like ProPage or another)...
    - routines of "clipping", to draw an object even if it's not completely
      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 suggestions and collaboration.
```

1.303 Possible future enhancements

```
    The following improvements will be made if the need is felt and if I have
    time to do them, (and also if I'm brave enough!)
    - choice of icon created by the program at the time of a save,
    - improvement of error message text, often vague,
    - improvement of function MAP (visualization and choice of macros already
      defined),
    - marking of a site in the diagram to find quickly, (practical in a
      large diagram),
    - requester to allow the choice of an object when there has to be a
      superposition,
    - writing an internal zoom function, (I use the freeware Lupe of
      Frank Toepper),
```

```
  - writing a library editor,
  - and more still... (see HELP!)

If you wish to translate this documentation or well the file
  catalogues, thank you me to communicate them so that they are
  distributed with the program.
```

1.304 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

The swedish catalog has been written by Henrik Isaksson:
  E-mail: henisak@algonet.se

The finnish catalog has been written by Joni Halme:
  E-mail: halmej@saunalahti.fi

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 into
English, (or deutsch?). Even translations of some items would be
appreciated.

Warning: The name of the nodes can't be changed: the program uses them
for the AmigaGuide online help.
```

1.305 AmiCAD2META

```
AmiCAD2META is a program that makes it possible to transform a file from
the AmiCAD format, (i.e. saved using AmiCAD), into a special format
used by the MetaView program and the library amigametaformat.library.

These two programs were written by Henk Jonas and make it possible
to save AMF files with the vectorial formats supported by this library.
```

For the moment the following formats are supported:
- WMF
- DR2D
- CGM
- GEM
- EPS
- AI
- HPGL
- ILBM

It will be necessary for you to obtain these programs on Aminet to be able to use them, (gfx/conv/MetaView, util/dtype/DT_MetaView, util/libs/amf_library), these not being distributed with AmiCAD.

You can also write in Henk Jonas to have precise details:
E-mail:subvcbhd@calvados.zrz.TU-Berlin.DE

To use AmiCAD2Meta you can use script ARexx Conv2Meta or directly call AmiCAD2Meta in a Shell window, the syntax of this program follows:

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

The first argument is obligatory (FROM), it specifies which file is to be treated. It must be a file created beforehand when saving in AmiCAD. Key word FROM can be omitted.

Example:
```
    AmiCAD2Meta Work:AmiCAD/Schemes/Logo
    AmiCAD2Meta FROM Work:AmiCAD/Schemes/Logo
```

The second argument specifies the name of destination AMF file, (with the Meta format). If it is not given, no conversion will happen. That can however be useful to check a file. If this destination file already exists, use the option FORCE so that it is overwritten.

Example:
```
    AmiCAD2Meta...  TO...  FORCE
```

The argument LIBS makes it possible to specify where the symbol libraries used by AmiCAD are located.

Example:
```
    AmiCAD2Meta...  LIBS Work:AmiCAD/Libraries
```

Argument VERBOSE makes it possible to display a certain amount of information on the screen whereas the QUIET argument makes it possible to not have any display, (useful in an ARexx script).
These last two arguments must be naturally be used one or the other, not both.

Example:
```
    AmiCAD2Meta...  VERBOSE
```

Last argument (PENWIDTH) makes it possible to widen (possibly) the features by the specified multiple. If this argument is not given the features are saved with the width present in the diagram, if not their width is multiplied by the value of this argument. Many programs do

```
    not utilise the width of the features but always traces them at the same
    width, (Wordworth using formats CGM or GEM, AI with ProPage).

  Example:
      AmiCAD2Meta...  PENWIDTH=2

  Example of call:
      AmiCAD2Meta Work:AmiCAD/Schémas/Test TO Work:MetaView/AmiCAD/Test.amf LIBS  ↩
         Work:AmiCAD/Bibliothèques
      AmiCAD2Meta Work:AmiCAD/Schémas/Test VERBOSE
```

1.306 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@fnac.net
```

1.307 Index

CHR
CLIPPATH
CLIPUNIT
CLOSE
COL
CONVERT
COORDS
COPY
CURSMODE

D

DATE
DEF
Definition of functions
DELARRAY
DELETE
DEVPINS
DIMARRAY
DIMSHEET
Distribution
DRAW
DRAWMODE

E

EDIT
ELLIPSE
ENDCOL
ENDLINE
EXEC

F

FILENAME
FINDLAB
FINDLINE
FINDOBJ
FILEPART
FINDPART
FINDREF
FINDVAL
FIRSTSEL
FONTNAME
FONTSIZE
FOR
Future

G

GETARRAY
GETCOLOR
GETDEVS

READCONV
READDEF
READDEV
READMAP
READTEXT
REMLIB
REQFILE
REQLIST
REQSHEET
REQUEST
RESET
REXXPORT
ROTATE
Running the program

S

SAVEIFF
SAVE
SAVEALL
SAVECLIP
SAVECOPY
SAVEICON
SAVEKEYS
SAVEPREF
SCREEN
Scripts
SCRMODE
SECURITY
SELECT
SELFILE
SETARRAY
SETCOLOR
SETDEV
SETFILL
SETGRID
SETLABEL
SETORTHO
SETPINS
SETREF
SETSCALE
SETTEXT
SETVAL
SGN
SHEET_HEIGHT
SHEET_WIDTH
SHEIGHT
SNAPGRID
STARTUP
STOBACK
STOFRONT
STR
SWIDTH
SYMMETRY

T

TEST
Thematic ARexx functions
TIME
TITLE
TOOLBAR
TRIANGLE
TXHEIGHT
TXWIDTH
TYPE

U

UNGROUP
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