AmiCAD

FLORAC Roland

COLLABORATORS

	TITLE : AmiCAD		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	FLORAC Roland	August 10, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmiCAD

1.1 Sommaire

```
                   -------------------------------------
    AmiCAD Version 1.4  October 11 1998
   -------------------------------------

    Warning: This guide is an incomplete translation of the
    french guide. The latest ARexx functions may not be present.
```

1.2 Distribution

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

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

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

You can reach me by sending your E-mails to the address
roland.florac@fnac.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.

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

```
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.

    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,
    (or AMICAD0, AMICAD1... AMICAD9 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. If the program is launched several times, the port is called AMICAD0 by the second task, then AMICAD1 for the following one, and so on until AMICAD9.

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("scale~horizontal")       Warning: space "solid"
   (ALT-SPACE) between scale and horizontal.
```

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("scale~vertical")      Warning: space "solid"
    (ALT-ESPACE) between scale and vertical.
```

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).

  See also: functions
              LOADPREF
              ,
              SAVEPREF
              .

    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: functions
              LOADKEYS
              ,
              SAVEKEYS
              ,
              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:

```
    HSCALE(FIRSTSEL, 2) : VSCALE(FIRSTSEL,2)
```

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

ASC
 returns an ASCII code

ASK
 ask for a string from the user

BLINK
 draws and clears an object

CALL
 calls an ARexx script

CHR
 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

COPY
 copy one or more objects into memory

DATE
 returns the actual date

DEF
 defines a new function

DELETE
 removal of an object

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

EXEC
 interpret a string of characters

FILENAME
 name of complete file, (including path)

FILEPART
 name of file

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

GETDEVS
 number of gates in a circuit

GETPART
 choose component to use

GETREF
 read the reference of a component

GETVAL
 read the value or some type of component

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

LEN
 length of the character string

LIBSPATH
 specify drawer containing the symbol libraries

LINE
 number of the line where to locate a object

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

LOADKEYS
 load a macro file

LOADLIB
 load a symbol library

LOADPREF
 loading of a file of preference

LOCK
 lock user input

MACRO
 calls a programmed key sequence

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

MEASURE
 return the dimensions of a window

MODIF
 test for modification

MOVE
 move an object

NBSHEET
 number of diagrams present in memory

NEW
 open a new window

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

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

REQSHEET
 choose a diagram

REQUEST
 display a message with a choice of YES/NO

RESET
 reinitialization of variables

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

SAVEKEYS
 save macros

SAVEPREF
 save file preferences

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

SELFILE
 selection of a diagram by its name

SELSHEET
 selection of a diagram by its index

SETCOLOR
 set the screen colour

SETDEV
 choose a circuit or gate of a component

SETGRID
 set the grid step

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

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

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

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

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

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

UNGROUP
 deletes the links of a group

WRITE
```

```
 place 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

DELETE
 removal of an object

EDIT
 call the edit requester

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

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

LIBSPATH
 specify the drawer containing the symbol libraries

LINE
 number of the line where an object is located

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

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

SETDEV
 specify the circuit or a gate of a component

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

LOADKEYS
 load a file of macros

LOADPREF
 load a file of preferences

SAVECOPY
 choose to save backup copies

SAVEICON
 choose to create icons for saved files

SAVEKEYS
 save macros

SAVEPREF
 save preferences

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
```

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

LOCK
 lock user input

MESSAGE
 display a message

PICKOBJ
 choose an object using the mouse

REQFILE
 choose a file

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

```
CALL
 calls an ARexx script

DATE
 returns the actual date

DEF
 defines a new function

EXEC
 interpret a string of characters

FINDOBJ
 look for an object

FINDPART
 look for a component

HELP
 displays an AmigaGuide help

MACRO
 calls a programmed key sequence

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

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

MEASURE
  return a dimension of window

MODIF
 test for modification

NBSHEET
 number of diagrams present in memory

NEW
 open a new window

OPEN
 open a diagram

PRINT
 print a diagram

REQSHEET
 choose a diagram

SAVEIFF
 save in IFF format

SAVE
 save a diagram

SAVEALL
 saves the document in the Undo cache

SELFILE
 selection of a diagram by its name
```

SELSHEET
selection of a diagram by its index

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

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

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 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
```

1.123 ARexx function ASK

```
                  This function has been suppressed in version 2.0
    Use one of the
              ASKNUM
               or
              ASKTEXT
               functions instead.

    It can be emulated with the following definition:
    DEF ASK(T)=ASKTEXT(T,"")

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

1.124 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
                .
```

1.125 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
                .
```

1.126 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
                .
```

1.127 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
                .
```

1.128 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)'
```

1.129 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
```

1.130 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")
```

1.131 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.

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
              .
```

1.132 ARexx function CLOSE

```
                  CLOSE(window)

  This function causes the closing of the window of specified index.  Each
  window has a different index, starting with index 0 (see
              SELSHEET
              ).
  The returned value corresponds to the number of windows remaining present
  in memory.

  Use command
              MENU
              ("Quit") to close all the windows and to end the
  program.

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

```
See also:
            OPEN
            ,
            LOAD
            .
```

1.133 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
            .

See also:
            LINE
```

1.134 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 1 if the selected object is well
    component, if not it is one 0.

  See also: menu Drawing/
                Alternate symbol
```

1.135 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 whereas 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
                .
```

1.136 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.
```

1.137 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
              .
```

1.138 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
                .

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

1.139 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.

    Recall: The windows are of the SuperBitmap type, with the result that
    their apparent surface can be smaller than their useful surface area,
    (it's enough to look at the state of the vertical and horizontal sliders
    to see if this one is partially hidden or not).  This type of window
    allows very fast operations on the drawing when the drawing is larger
    than the screen, however it can be greedy in CHIP memory, if you specify
    a large size.  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.

    Note:  The width is always brought back to a value which is a multiple
    of 16 in order to ensure correct saving of the drawing by using the menu
```

```
             Projet/Save format IFF
             .

    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
```

1.140 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.
```

1.141 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.
```

1.142 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.

  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.
```

1.143 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)

See also:
             ARC
```

1.144 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 -)
```

1.145 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
```

1.146 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
```

1.147 ARexx function FINDOBJ

```
                  FINDOBJ(first_object,x,y)

    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 coordinates must match an extremity for lines, or the point
    where the cursor was at the moment the object was placed.

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

1.148 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)#?")'
```

1.149 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
```

1.150 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
```

1.151 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
```

1.152 ARexx function FONTNAME

```
                    FONTNAME(x)

    Return the name of the font used in the current window.  The argument
    can take any value.  Use preferably argument
                SELSHEET
                (-1), for later
    compatibility.

  Example:
      FONTNAME(SELSHEET(-1))   returns topaz.font (for example)

  See also:
                FONTSIZE
                .
```

1.153 ARexx function FONTSIZE

```
                  FONTSIZE(x)

    Return the size of the font used in the current window.  The argument
    can take any value.  Use preferably argument
                SELSHEET
                (-1), for later
    compatibility.

  Example:
      FONTSIZE(SELSHEET(-1))   returns 11 (for example)

  See also:
                FONTNAME
                .
```

1.154 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
              .
```

1.155 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
              .
```

1.156 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)

  See also:
              READDEV
              ,
              SETDEV
              .
```

1.157 ARexx function GETPART

```
                  GETPART("name_component")

    Makes it possible to choose the component which will be drawn using
    the menu "Drawing/Place component".  The name of the component can appear
    in upper or lower case.
    The function returns 1 if the component was found, 0 if not.

  Examples: GETPART("RESISTANCE")
      IF(GETPART(ASK("Component?")),MENU("Place~component"),0)

    Note, interesting information:  The name of the component can be
    specified by its first letters, the first component whose name begins
    the same as the argument which will be selected.  Thus in the second
    example it is enough to state the letter R in the requester to choose
    a resistance as the component, (if the library Symbols_AmiCAD is the
    first of the list of libraries loaded).

  See also:
              LOADLIB
              , menu
              Drawing/Place component
```

1.158 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
```

1.159 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
```

1.160 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
               .
```

1.161 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
               .

  See also:
               WIDTH
```

1.162 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")
```

1.163 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
                .

  See also:
                VSCALE
                ,
                SETSCALE
                .
```

1.164 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)
```

1.165 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.

  See also:
                RESET
```

1.166 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
```

1.167 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.
```

1.168 ARexx function LEN

```
    LEN("string")

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

1.169 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 */
```

1.170 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
                .

  See also:
                COL
                ,
                COORDS
                .
```

1.171 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
```

1.172 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
```

1.173 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
```

1.174 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
                .
```

1.175 ARexx function LOADKEYS

```
                    LOADKEYS("macro_file")

    This function makes it possible to load the key combinations defined in
    the file passed in the argument.  This file must be saved beforehand by
    the menu
                Preferences/Keys/Save
                 or the function
                SAVEKEYS
                .

  Example: LOADKEYS("Keys AmiCAD")

  See also:
                MAP
                ,
                Mode direct
```

1.176 ARexx function LOADLIB

```
                    LOADLIB("name_library")

    This function makes it possible to load a library of symbols into
    memory, it returns 1 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
                .
```

1.177 ARexx function LOADPREF

```
              LOADPREF("nom_fichier")

Load the specified prefs file. This file must have been previously
created using the menu
              Preferences/Save
              .

See also:
              SAVEPREF
```

1.178 ARexx function LOCK

```
                  LOCK(x)

  Lock the window of the specified index (see
              SELSHEET
              ).
  If x is -1 all the windows are locked.  i.e. 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.
  Certain functions, like the choice of a file, saving, loading, printing,
  involve a temporary locking of all the windows.
  You may generally find it beneficial to lock all the windows for a
  particular condition (LOCK(-1)).
  To lock only the active window, use the following: LOCK(SELSHEET(-1))

  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
              .
```

1.179 ARexx function MACRO

```
                  MACRO(x)

  Carry out the macro defined on one of the ten function keys F1 (x=1) to
  F10 (x=10).
  This function is especially practical to carry out calls of macros
```

from other macros. I use it in connection with the menu

Macros/Direct
.
Thus the key F4 can for example, call the macro defined for the key F5, (MACRO(5)).

Returned value: result of the macro calculation.

Note that the type of the result of macro can be unspecified: numerical or character string.

1.180 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).
Only single character keys are definable (for the moment): you cannot define, using this function, either the function 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"")")
```

See also:

READMAP
,
UNMAP
, definition of
Function keys
.

1.181 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 entered).

  See also:
                UNMARK
                ,
                MARKZONE
                .
```

1.182 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
                .
```

1.183 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?).

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

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

    Note: The menu title has to match the localized strings, the ARexx
    scripts using this function have to be translated into foreign langages
    if their catalogs exist.
```

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

1.184 ARexx function MESSAGE

```
  MESSAGE("string")

  Display a requester containing the specified message.  This text can
  contain from one to thirteen lines separated by linefeeds.

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

Returned value:  0.
```

1.185 ARexx function MEASURE

```
                MEASURE(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
```

1.186 ARexx function MODIF

```
                MODIF(window)

  Allows you to know if a window containing a diagram has been modified
  since it was last saved.
  The argument can be positive or null, like what can be returned by the
  function
              SELSHEET
```

```
            :  it is information concerning this
window which is then returned.
If the argument is negative, it is for the current window that
information is returned.

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.
```

1.187 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
            .

No value is returned by this function.
```

1.188 ARexx function NBSHEET

```
  NBSHEET(x)

  Return the number of diagrams present in memory.  This counting is
  carried out on whole or part of the windows, according to the value of
  the argument:
equal 0:  counts only the hidden diagrams
equal 1:  counts only the open diagrams (not hidden or iconified)
equal -1: all the diagrams are counted

Example:
    NBSHEET(-1)-NBSHEET(0)  'returns the number of windows iconified
```

1.189 ARexx function NEW

```
NEW("title")

Cause the opening of a new window, the name of this window is that
passed in argument.  If the argument is a null string, the window takes
```

```
    the default "NoName" name.

    Returned value:  1 if succeeded, 0 if not.
```

1.190 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
                .
```

1.191 ARexx function OBJECTS

```
                    OBJECTS(window)

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

   See also:
                SELSHEET
```

1.192 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 enough 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 0 if all went well, an error code (non null value)
    if there was a problem.

  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
                .
```

1.193 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.

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

              SYMMETRY
              ).

  See also:
              INPUT
```

1.194 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
```

1.195 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
.

1.196 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
.

1.197 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.

1.198 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.

1.199 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.

  See also:
                GETPART
                ,
                PARTNAME
```

1.200 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
                .

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

1.201 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
```

1.202 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
                .
```

1.203 ARexx function READMAP

```
                    READMAP("key combination")

    This function returns a string with the definition associated to the
    argument.
    If there is no definition an empty string is returned.

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

  See also:
```

```
              MAP
              ,
              UNMAP
```

1.204 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
```

1.205 ARexx function REMLIB

```
                  REMLIB("name_library")

   Remove the specified library from memory.

   Warning:  all the elements which result from it's use will be removed
   from the diagram, (without warning).

   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
```

1.206 ARexx function REQFILE

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

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

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

1.207 ARexx function REQSHEET

```
                REQSHEET("title")

  This function makes it possible to choose, using a file requester, a
  diagram among those which are loaded in memory.
  The title is displayed in the first line, it should not include a
  linefeed.
  The number of the selected diagram, (1 to x, according to the number of
  diagrams present in memory), is returned.
  If the user presses on the right mouse button or if a problem occurs, a
  value equal to or lower than zero 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 and the number of the selected button
  is returned.
```

1.208 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.
```

1.209 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
```

1.210 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
```

1.211 ARexx function SAVEIFF

```
                    SAVEIFF("file name")

    The current document is saved using the IFF format.
    A 0 is returned if it worked.

  See also:
                Project menu/Save IFF
```

1.212 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
              .
```

1.213 ARexx function SAVEALL

```
                  SAVEALL(window)

    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
              .

    If the argument is positive or null, the selected window is
    selected, (see
              SELSHEET
               function), and the operation is done in it.
    If the argument is negative, (usualy -1), the operation is done in
    the current window.

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

  See also:
              SELSHEET
```

1.214 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
              .
```

1.215 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.

    To read the state of the menu without modifying it, use the command
    SAVECOPY(-1).

  See also:
                SAVEICON
                .
```

1.216 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.

    To read the state of the menu without modifying it, use the command
    SAVEICON(-1).

  See also:
                SAVECOPY
                .
```

1.217 ARexx function SAVEKEYS

```
                    SAVEKEYS("nom fichier")

    This function makes it possible to save the key combinations currently
    defined, (combinations of keys and function keys), in the file passed
    in argument.
    These definitions could then be reloaded by the menu

                Preferences/Keys/Load file
                 or the function
                LOADKEYS
                .

  See also: MAP,
                Mode direct
```

1.218 ARexx function SAVEPREF

```
                 SAVEPREF("filename")

    This function makes it possible to save the current preferences under the
    specified name.
    It returns 0 if all went well, an error code if not.

  See also:
              LOADPREF
```

1.219 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 eight 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
            .
```

1.220 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
```

1.221 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.
```

1.222 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...).  There can be to 13 buttons.

    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")
```

1.223 ARexx function SELFILE

```
                SELFILE("file_name")

  This function makes it possible to choose the active window, by
  specifying its name.  The specified name can be the complete name, or
  the name of the file, without the path.
  If several windows have the same name, the first found window is
  selected.
  However the selected window is not moved to the foreground, use the
  function
              WTOFRONT
               for that.
  The same if the window is iconified or hidden, it will remain so.
  This is useful to read a line or some characters of a window without
  "awaking it".

  The returned value is positive or null if the window was found, (it
  corresponds to the value that would have been returned by the function

              SELSHEET
              (-1)), if not it is negative.

    Examples:
  SELFILE("RAM:text")
  SELFILE("texte")

  These two examples allow the selection of the same window, (RAM:text).
```

1.224 ARexx function SELSHEET

```
                SELSHEET(window)

  This function makes it possible to choose the active window or to find
  the index of the current window.
  If the argument is positive or null, the specified window is selected
  and becomes the active window.
  However the selected window is not brought to the foreground, use the
```

```
function
            WTOFRONT
             for that.
The same if the window is iconified or hidden, it will remain so.

The function always returns the index of the active window before it's
execution.
If the index passed in argument is negative, only this value is returned.
```

1.225 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 7 inclusive, (the screen has eight 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.
```

1.226 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
            .

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
            .
```

1.227 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.
```

1.228 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
```

1.229 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
```

1.230 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.
```

1.231 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
```

1.232 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
```

1.233 ARexx function SGN

```
    SGN(number)

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

1.234 ARexx function SHEIGHT

```
                  SHEIGHT

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

  See also:
              SCREEN
              ,
              SWIDTH
              ,
              SCRMODE
```

1.235 ARexx function STOBACK

```
                  STOBACK

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

  See also:
              STOFRONT

              WTOBACK

              WTOFRONT
```

1.236 ARexx function STOFRONT

```
                 STOFRONT

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

 See also:
             STOBACK

             WTOBACK

             WTOFRONT
```

1.237 ARexx function STR

```
   STR(number)

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

1.238 ARexx function SWIDTH

```
                 SWIDTH

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

 See also:
             SCREEN
             ,
             SHEIGHT
             ,
             SCRMODE
```

1.239 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
```

1.240 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.
```

1.241 ARexx function TIME

```
                  TIME(seconds)

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

 Examples:
     TIME(0) return 23:24
     TIME(1) returns 23:24:27

 See also:
              DATE
```

1.242 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
```

1.243 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
```

1.244 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
```

1.245 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 feature
      - equal to 21:  it is a personalized line (unspecified width)

    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 CONNECTER(O) = IF ((TYPE(O)==11) | (TYPE(O)==12), 1,0)

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

1.246 Arexx function UNGROUP

```
                UNGROUP(group)

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

  Returns the number of objects that were in the group.

See also:
            GROUP
            .
```

1.247 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.
  If the component had these two elements, the function returns 2, if it
  had one of them, it returns 1 if not it returns 0.
  If the specified object is not a component returned value is equal to -1.

Examples:
    UNLINK(FIRSTSEL)
    UNLINK(PICKOBJ("Click on a component"))

See also:
            LINKVAL
            ,
            LINKREF
```

1.248 ARexx function UNLOCK

```
                    UNLOCK(window)

    Cancel the locking of a window, initiated by function
                LOCK
                .
    If the argument is positive or null, only the window having the
    specified index is unlocked, if this argument is negative (-1), all
    the windows are unlocked.

  See also:
                SELSHEET
                .
```

1.249 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).

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

  See also:

                MAP
                ,
                READMAP
```

1.250 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
```

1.251 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
```

1.252 ARexx function VERSION

```
    VERSION(arg)

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

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

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

1.253 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
                .

  See also:
```

HSCALE
,
SETSCALE
.

1.254 ARexx function WHEIGHT

```
                  WHEIGHT(window_index)

   Return the useful height of the specified window.
   The index of the window corresponds to the value returned by function

              SELSHEET
              .
   If the argument is -1, it is the current window's index which is used.

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

   Remember: The windows are of the SuperBitmap type, which is to say that
   their visible surface may not be equal to that which they could take,
   if they are larger than the screen.

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

1.255 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
```

1.256 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
```

1.257 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
```

1.258 ARexx function WRITE

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

    Place a text at the specified coordinates.
```

The current horizontal and vertical scales are used (see SETSCALE, ROTATE, SYMMETRY).

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

See also: READTEXT.

1.259 ARexx function WTOBACK

```
WTOBACK(window)
```

Send the window of specified index or the current window (argument equal to -1) to the background.

See also: SELSHEET, WTOFRONT, STOBACK, STOFRONT

1.260 ARexx function WTOFRONT

```
WTOFRONT(window)
```

Send the window of specified index or the current window (argument equal to -1) to the foreground.

See also: SELSHEET, WTOBACK

1.261 ARexx function WWIDTH

```
WWIDTH(window_index)
```

Return the working width of the specified window. The code of the window corresponds to the value returned by function

SELSHEET
.
If the argument is -1, it is the current window which is treated.
The returned value corresponds to the height of the document, in pixels.

Remember: The windows are of the SuperBitmap type, which is to say that their visible surface may not be equal to that which they could take, if they are larger than the screen.

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

1.262 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
```

1.263 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).
```

1.264 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
```

1.265 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.
```

1.266 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"))
```

1.267 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...
```

1.268 History

```
              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.
```

1.269 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.
```

1.270 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.

1.271 Translation

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

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

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

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

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

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

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

I am looking for guys that could help me to translate the guide file 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.

1.272 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
```

1.273 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

1.274 Index