MicroRexx

COLLABORATORS

	TITLE : MicroRexx		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 7, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MicroRexx

1.1 MicroRexx 2.0.2 © 1997-2000 by FR-SoftWorks

1.2 About ARexx

What is ARexx?

ARexx is a powerfull scripting language which is used for controlling Amiga DOS and other applications.

It is part of the operating system since Workbench 2.0.

How do I write ARexx scripts?

Scripts are written in plain ASCII and can thus be created using any text editor.

How do I execute ARexx scripts?

Perform the following steps to start an ARexx script:

1 Ensure that the ARexx interpreter RexxMast is running.

2 Open a shell window.

3 Ensure that your path contains SYS:REXXC.

4 Type "rx scriptname" to start a script.

Note: If the name of your script ends up with .rexx, you can start it by typing "rx scriptname", omitting the extension ".rexx".

String problems

If only the first word of a string is used, try this instead of the string:

```
'"' || string || '"'
```

1.3 Installing MicroRexx

MicroRexx requires at least Amiga OS Release 2.1 and 512 KB RAM.

Note: The "locale.library" must be present. Otherwise MicroRexx will hang until the library is available. (This is a "feature" of Blitz Basic.)

To activate MicroRexx every time your Amiga starts up, perform the following steps:

1 Add the Tooltype 'DONOTWAIT' to the program icon.

2 Drag the program icon into your »WBStartup« drawer.

1.4 introduction

MicroRexx extends the capabilities of the ARexx interpreter "RexxMast" and allows you to interact with the user via GadTools gadgets and Intuition requesters.

With MicroRexx, you can open up to 1,024 windows at the same time, define the text font and style of each of them and handle gadgets very easily.

Starting with version 2.0.0, MicroRexx does also support GadTools menus and provides a lot of mathematical and string manipulation functions. Encryption and locale support are also available.

The program will close all windows automagically when it is shut down.

Note: You MUST NOT quit MicroRexx if an ARexx script using one of its windows is still running, because the script would not be able to terminate in that case.

The ARexx port provided by the program is also called 'MicroRexx'.

» To execute an ARexx script, open a CLI window and type "rx <scriptname>".

1.5 Overwiew

MicroRexx has the following features:

o up to 1,024 windows
o different fonts and text styles
o system and ASL requesters
o GadTools gadgets
o Locale support
o number formatting
o system information
o simple encryption
o GadTools menus
o mathematical functions
o string manipulation

1.6 Project

About	Displays version number and copyright notice.
Quit	The program terminates immediately.

1.7 Copyright & liability

Copyright (C) 1997-2000 by FR-SoftWorks.

The author shall not be liable for any damages caused by the usage of this program.

This program is placed in the public domain and may be freely distributed. (Ensure that you always distribute the whole archive, however.)

Send me a post card if you use MicroRexx, please. Write bug reports and / or suggestions to:

Frank Reibold
Ottberger Weg 13
D-31737 Rinteln

GERMANY

eMail: frank@reibold-online.de

Have fun!

1.8 History

1.0.0   First public release.

```
1.0.1   Willem Schaaij reported a lot of Enforcer hits (thank you for your
        support!). The Enforcer hits were caused by:

        o  the ARexx message handler by processing anything without checks
           for illegal message pointers,

        o  the OPENWIN command by using wrong window IDs internally.

        The Enforcer hits have been fixed.

1.0.2   Don Cox reported that the title bar of the main window was allways
        11 pixels, regardless of the font size.

        o  MicroRexx uses the entry BarHeight of the Workbench’s Screen
           structure to determine the height of the title bar.

        The main window is font sensitive now.

2.0.0   Support for GadTools menus has been added. Math functions and string
        manipulation have been introduced.

        All commands and functions are case insensitive now.

        The event handler loops of both the scripts and MicroRexx have been
        rewritten.

2.0.1   Kevin McCarthy reported that MicroRexx 2.0.0 crashed on his Amiga
        500 with Kickstart 3.1 and Workbench 1.3.

        The ISLOCALE command used the IsLocale function of BlitzBasic which
        seems to be buggy.

        o  MicroRexx tries to open the current locale settings instead of
           using the IsLocale function.

2.0.2   Jörg Rebenstorf reported that the QUIT command of MicroRexx 2.0.1 threw
        Enforcer hits.

        MicroRexx used a special function for removing its own GadTools
        menu.

        o  The function mentioned above is no longer called.

        Support for Amiga OS 3.5 has been added.
```

1.9 Examples

This chapter mainly contains information on how to write event handlers for MicroRexx scripts.

```
   How to use gadgets

   How to use menus
```

Have a look at Misc.rexx for an overview of all other commands.

1.10 How to use menus

To learn how MicroRexx works, have a look at the following example (do not enter the line numbers):

```
 1    /* Menu.rexx */
 2    options results
 3    address 'MicroRexx'
 4    'OPENWIN' 10 10 300 100 'MicroRexx Demo Window'
 5    w = result
 6    'MENU' w 0 "Project"
 7    'ITEM' w 0 0 0 "About" "?"
 8    'ITEM' w 64 0 1
 9    'ITEM' w 0 0 2 "Quit" "Q"
10    'MENUON' w
11    do forever
12      'MENUHIT' w
13      mh = result
14      if mh = 0 then do
15        'ITEMHIT' w
16        ih = result
17        if ih = 0 then do
18          'REQUEST "MicroRexx Demo" "Select QUIT to exit." "Continue"'
19        end
20        if ih = 2 then do
21          signal 1
22        end
23      end
24    end
25    1:
26    'MENUOFF' w
27    'KILLMENU' w
28    'CLOSEWIN' w
29    exit
```

Line #	Remarks
1	This comment is required by the ARexx interpreter "RexxMast".
2	This line ensures that the ARexx interpreter puts the result of all commands into a system variable called "result".
3	The ARexx port MicroRexx is used and the commands provided by MicroRexx are available from now on.
4	A new window is opened.
5	The variable "w" is used to store the window handle.
6	A menu title is declared.
7	A menu item is declared.
8	A menu barlabel item is declared.
9	A menu item is declared.
10	The menu is drawn and activated.
11	The ("never ending") event handler loop starts here.
12	Store menu number, if any.
13	The variable mh is used to store it.

14	Has menu #0 been selected?
15	Yes. Store menu item number, if any.
16	The variable ih is used to store it.
17	Has item #0 been selected?
18	Yes. The About Requester is displayed.
19	End of If statement.
20	Has item #2 been selected?
21	Yes. Raise exception #1 --> go to label "1:"
22	End of If statement.
23	End of If statement.
24	End of loop.
25	Declaration of Label 1. The Event handler for exception #1 starts here. (This piece of code will also be executed when the loop ends.)
26	The menu is disabled. (This is required by the next command. Call it before altering menus or menu items respectively.)
27	The menu is destroyed. (This is required by the next command.)
28	The window #w is closed.
29	The script terminates.

1.11 How to use gadgets

To learn how MicroRexx works, have a look at the following example (do not enter the line numbers):

```
 1    /* Window.rexx */
 2    options results
 3    address 'MicroRexx'
 4    'OPENWIN' 10 10 300 100 "Window"
 5    w = result
 6    'BUTTON' w 0 30 30 30 30 "QUIT" 2
 7    'STRING' w 1 100 50 80 16 "»" 1 5
 8    'GADGETSON' w
 9    'GADGETHIT' w
10    do forever
11      'GADGETHIT' w
12      g = result
13      say g
14      if g = 1 then do
15        'STRINGRESULT' w 1
16        say result
17      end
18      if g = 0 then do
19        signal 1
20      end
21    end
22    1:
23    'GADGETSOFF' w
24    'KILLGADGETS' w
25    'CLOSEWIN' w
26    exit
```

Line #   Remarks

```
 1         This comment is required by the ARexx interpreter "RexxMast".
 2         This line ensures that the ARexx interpreter puts the result of
           all commands into a system variable called "result".
 3         The ARexx port MicroRexx is used and the commands provided by
           MicroRexx are available from now on.
 4         A new window is opened.
 5         The variable w stores the window handle which must be a number
           between 100 and 1,024, else something went wrong. The window
           handle is required by the following commands.
 6         A command button "QUIT" is created.
 7         A string gadget is created.
 8         The gadgets of window #w are drawn and can be used.
 9         The gadget handler is activated.
10         The ("never ending") event handler loop begins here.
11           The gadgets are handled.
12           The variable g stores the number of the selected gadget (-1 means
             that none has been selected).
13           The variable g is displayed in the current CLI window.
14           Has gadget #1 been selected?
15             The content of the string gadget (our gadget #1) is read.
16             The result is displayed in the current CLI window.
17           The if command ends here.
18           Has gadget #0 been selected?
19             This command raises the exception #1 --> go to label "1:"
20           The if command ends here.
21         The loop ends here.
22         This code will be executed in case of the exception 1 or if the
           program terminates.
23         The gadgets are disabled.
24         The gadgets are removed. This is necessary if a gadget is to be
           changed or if you want to close the window.
25         The window #w is closed.
26         The ARexx script ends here.
```

1.12 Mathematical functions

The following mathematical functions are available:

Function	Arguments	Result
ABS	Number	Absolute value of Number
ACOS	Number	Inverse Cosine of Number
ASIN	Number	Inverse Sine of Number
ATAN	Number	Inverse Tangent of Number
COS	Number	Cosine Number
D2R	Number	Angle Number converted from degrees to radians
EXP	Number	E to the power of Number
FRAC	Number	Fractional part of Number
HCOS	Number	Hyperbolic Cosine of Number
HSIN	Number	Hyperbolic Sine of Number
HTAN	Number	Hyperbolic Tangent of Number
INT	Number	Nearest integer value below Number
LOG	Number	Log base E of Number
LOG10	Number	Log base 10 of Number

R2D	Number	Angle Number converted from radians to degrees
RND	[Number]	Random value betwen 0 and Number (Number defaults to 1)
SGN	Number	Sign of Number
SIN	Number	Sine of Number
SQR	Number	Square root of Number
TAN	Number	Tangent of Number

1 All trigonometrical functions expect Number to be an angle stated in radians.

1.13 commands

Notes:

o The commands are stated in upper case, but you can mix all cases (pRiNt is the same as PRINT).

o All parameters have to be supplied in the order stated below.

o Optional parameters are stated within brackets ([...]) which must be omitted.

o All arguments must be seperated by blanks.

o Character arguments should be quoted using apostrophes. You must use quotes if the character string contains spaces.

Example: ’PRINT’ w ’"This is just a demonstration."’

o Results are passed to the ARexx variable result if the command "options results" is given before switching to the ARexx port of MicroRexx.

If the result is "ERROR", then something went wrong.

Since version 2.0, MicroRexx does also provide mathematical functions .

ABOUT

This command has no arguments.

ABOUT displays version number and copyright notice.

AGA

This function has no arguments.

AGA returns -1 if the AA chipset has been found, otherwise 0.

BOX

This command has the following arguments:

```
  window           Window handle.
  x1               \ First coordinate.
  y1               /
  x2               \ Second coordinate.
  y2               /
```

BOX draws a rectangle.

BUTTON

This command has the following arguments:

```
  window           Window handle.
  id               The identification number of the button.
  x                \ Coordinates in pixels.
  y                /
  width            \ Width and height of the button.
  height           /
  text             The text to be displayed with the button.
  flags             1                  / left
                    2                 /  right
                    4 Text position: |   above
                    8                 \  below
                   16                  \ in
```

BUTTON creates a command button.

CHECKITEM

This command has the following arguments:

```
  window           Window handle.
  menu#            Number of the menu.
  item#            Number of the item.
  check            if 1: menu item is checked
                   if 0: menu item is not checked.
```

Note: The flag of the menu item must be 256 or 257. (See ITEM.)

CHECKITEM allows you to toggle checked menu items.

CIRCLE

This command has the following arguments:

```
  window           Window handle.
  x                \ Coordinates in pixels.
  y                /
  xr               X radius.
  yr               Y radius.
```

CIRCLE draws an ellipse.

CLOSEWIN

This command has the following argument:

window	Window handle.

```
Notes: (1) If gagdgets have been created, call KILLGADGETS before
           calling CLOSEWIN.
       (2) If menus have been created, call KILLMENU before
           calling CLOSEWIN.
```

CLOSEWIN closes a window.

CLRSCR

This command has the following argument:

window	Window handle.

CLRSCR clears the contents of a window.

CPU

This function has no arguments.

CPU returns the CPU type (for example 4 = 68040).

CRC32

This function has the following argument:

string	String to be processed.

CRC32 returns the CRC32 checksum of a string.

CRYPT

This function has the following argument:

string	The string to be processed. Encrypted strings will be decoded automatically.

CRYPT returns an encrypted or decoded string respectively.

ECLOCK

This function has no arguments.

ECLOCK returns the EClock frequency.

EDIT

This function has the following arguments:

```
  window           Window handle.
  [length]         The maximum length of the string to be edited (or 80).
```

EDIT allows you to edit a text within a window.

FONT

This command has the following arguments:

```
  window           Window handle.
  name.font        The name of the font (for example times.font).
  size             The size of the font (for example 13).
  [style]          Refer to the STYLE command for details, please.
```

FONT allows you to change the text font of a window.

GADGETHIT

This function has the following argument:

```
  window           Window handle.
```

GADGETHIT returns the number of the selected gadget or -1 if none has been hit.

GADGETSOFF

This command has the following argument:

```
  window           Window handle.
```

GADGETSOFF disables the gadgets of a window.

GADGETSON

This command has the following argument:

```
  window           Window handle.
```

GADGETSON draws and enables the gadgets of a window.

IFMT

This function has the following argument:

```
  n                An integer value.
```

IFMT returns a number formatted according to the Locale settings (1234 becomes 1,234 if "Great_britain" is the currently selected country, for instance).

INPUT

This function has the following arguments:

[Title]	This defines the title of the window (max. 20 characters are accepted).
[Question #1] [Question #2]	These are the questions which will be displayed in the window. Each string must not exceed 25 characters.
[default value]	This value will be the initial value of the requester.

INPUT returns the string you entered.

ISFPU

This function has no arguments.

Note: If you have a 68040 CPU and the "68040.library" has not been installed, ISFPU returns 0. (The FPU commands are not available in that case.)

ISFPU returns -1 if an FPU is installed, otherwise 0.

ISLOCALE

This function has no arguments.

ISLOCALE returns -1 if the "locale.library" is available, otherwise 0.

ITEM

This command has the following arguments:

window	Window handle.
flags	The Intuition flags used for menu items.
menu#	The number of the menu to be used. The menu must have been created using the command MENU before.
item#	The number of the item to be created, starting with 0.
[text]	The text of the menu item.
[key]	The shortcut associated with the menu item.

The following flags are available:

Flag	Description	Example
0	Creates a default menu item.	Test
64	Creates a barlabel item.	------
256	Creates a checkable menu item which is not checked.	Test

```
  257   Creates a checkable menu item which is checked.     ./Test
```

Note: The menu has to be activated using the MENUON command before using it.

ITEM allows you to declare menu items.

ITEMHIT

This function has the following argument:

```
  window           Window handle.
```

ITEMHIT returns the number of the selected menu item or -1 if none has been selected.

KILLGADGETS

This command has the following argument:

```
  window           Window handle.
```

KILLGADGETS removes the gadgets of a window. This command has to be used before closing a window and before altering a gadget.

KILLMENU

This command has the following arguments:

```
  window           Window handle.
```

```
Notes: (1)  Call MENUOFF before calling KILLMENU.
       (2)  You must call KILLMENU before closing the window unless no
            menus are used.
```

KILLMENU destroys the menu of the specified window.

LINE

This command has the following arguments:

```
  window           Window handle.
  x1               \ First coordinate.
  y1               /
  x2               \ Second coordinate.
  y2               /
  [colour]         Colour to be used (for example 2).
```

LINE draws a line.

LISA

This function has no arguments.

LISA returns version number of the LISA chip as a hexadecimal number (00=OCS, F7=ECS, F8=AA).

LOCALE

This function has the following arguments:

```
  catalog          Catalog file to be used (including path).
  no               # of the string to be returned.
  default          Default string if the catalog file is not available.
```

```
Notes: Your Amiga might crash if you use a string number which has not been
       defined in the catalog file yet.
       The default string is returned if the catalog file could not be
       found.
```

LOCALE returns a locale string.

LOCATE

This command has the following arguments:

```
  window           Window handle.
  x                \ New cursor position in pixels.
  y                /
```

LOCATE moves the text cursor to a new position.

MENU

This command has the following arguments:

```
  window           Window handle.
  menu#            the number of the menu to be created, starting with 0.
  text             The new menu title.
```

```
Notes: (1) Use ITEM to add menu items to the menu.
       (2) Use MENUON to activate the menus.
       (3) Use MENUOFF to deactivate the menus.
```

MENU declares a menu (title).

MENUHIT

This function has the following arguments:

```
  window           Window handle.
```

MENUHIT returns the number of the menu selected, or -1 if none has been selected.

MENUOFF

This command has the following argument:

window	Window handle.

```
Notes: (1) Call MENUOFF before altering menus or menu items, and call
           MENUON after that.
       (2) MENUOFF must be called before calling MENUKILL.
```

MENUOFF deactivates the menus of a window.

MENUON

This command has the following argument:

window	Window handle.

Note: Use MENU and ITEM to declare menus or menu items respectively.

MENUON draws and activates the menus of a window.

MFLOPS

This function has no arguments.

```
Note: The calculation needs about one second. No over program except
      MicroRexx should be running, otherwise the result will be of no use.
```

MFLOPS returns the MFLOPS of your FPU (numerical co-processor) or "No FPU"
if no FPU is installed.

MIPS

This function has no arguments.

```
Note: The calculation needs about one second. No over program except
      MicroRexx should be running, otherwise the result will be of no use.
```

MIPS returns the MFLOPS (million instructions per second) of your CPU.

NPRINT

This command has the following arguments:

window	Window handle.
text	Text to be displayed.

NPRINT displays a text and performs a line feed.

NTSC

This function has no arguments.

NTSC returns -1 on NTSC systems, otherwise 0.

OPENWIN

This function has the following arguments:

```
  x                \ Window position in pixels.
  y                /
  width            \ Window size in pixels.
  height           /
  title            Title of the window.
  [flags]             1 Add size gadget.
                      2 Allow window dragging.
                      4 Add depth gadget.
                      8 Add close gadget.
                     16 To be used with 1 and 1024.
                     32 To be used with 1 and 1024.
                    256 Create a backdrop window.
                   1024 GimmeZeroZero: Ensures that the display does not
                        exceed the window limits.
                   2048 Create a borderless window.
                   4096 Activate the window immediately.
```

OPENWIN opens a new window and returns its handle.

OS

This function has no arguments.

OS returns the OS version (for example 40).

PAL

This function has no arguments.

PAL returns -1 on PAL systems, otherwise 0.

POWER

This function has no arguments.

POWER returns the power supply frequency.

PRINT

This command has the following arguments:

```
  window           Window handle.
  text             Text to be displayed.
```

NPRINT displays a text.

QUIT

This command has no arguments.

QUIT terminates MicroRexx immediately after closing all windows. Ensure that
no other ARexx script is using MicroRexx before shutting it down.

REPLACE

This function has the following arguments:

string	String to be changed.
search	Search string.
replace	Replace string.

Example: 'REPLACE' '"This is a test."' '" "' '""' returns "Thisisatest.".

REPLACE searches for a string and replaces it by another one.

REQUEST

This function has the following arguments:

[title]	Title of the requester. If it is omitted, a default text will be used.
text	Text to be displayed (Use '|' to insert a line feed).
[gadgets]	The gadgets to be drawn. The gadgets must be seperated by vertical slashes ('|'). If the gadget definition is omitted, "Yes|No" will be used.

REQUEST creates a requester and returns the number of the selected gadget
or 0 if the right most has been hit.

REQUESTFILE

This function has the following arguments:

[title]	Title of the requester.
[pattern]	The pattern to be used (for example '.IFF').

REQUESTFILE opens an ASL file requester and returns the name of the selected
file.

RFMT

This function has the following argument:

n	A real value.

RFMT returns a number formatted according to the Locale settings (1234.56 becomes 1.234,56 if "Deutschland" is the currently selected country, for instance).

SHAPE

This command has the following arguments:

```
  window           Window handle.
  x                \ The coordinate of the brush to be displayed.
  y                /
  [file name]      Name of the file the brush is stored in. The file name
                   can be omitted if the brush has been loaded already.
```

SHAPE (loads and) displays a brush.

STRING

This command has the following arguments:

```
  window           Window handle.
  id               Identification number of the string gadget.
  x                \ Coordinates in pixels.
  y                /
  width            \ Size of the gadget.
  height           /
  text             Text to be displayed with the gadget.
  flags             1                  / left
                    2                 /  right
                    4 Text position: |   above
                    8                 \  below
                   16                  \ in
  length           Maximum length of the string to be edited.
```

STRING creates a string gadget.

STRINGRESULT

This function has the following arguments:

```
  window           Window handle.
  id               Identification number of the string gadget to be examined.
```

STRINGRESULT returns the content of a string gadget.

STYLE

This command has the following arguments:

```
  window           Window handle.
  style             0 plain
                    1 underlined
```

```
                      2 bold
                      4 italic
                      8 extended
                     64 colour
```

STYLE changes the text style of a window.

TICKS

This function has no arguments.

TICKS returns the Intuition ticks elapsed since startup.

VBL

This function has no arguments.

VBL returns the vertical blank frequency.

VBR

This function has no arguments.

VBR returns the address of the Vector Base Register as a hexadecimal number (for example 07035F7C).

WB

This function has no arguments.

WB returns the Workbench version (for example 3.1).

WBDEPTH

This function has no arguments.

WBDEPTH returns the depth of the Workbench screen.

WBHEIGHT

This function has no arguments.

WBHEIGHT returns the height of the Workbench screen.

WBWIDTH

This function has no arguments.

WBWIDTH returns the width of the Workbench screen.

1.14 Index