COLLABORATORS			
	TITLE : PGS.MSU		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PGS.MSU

1.1 PageStream3 Macros

PageStream3 Macros

PageStream features a full ARexx implementation with hundreds of commands to allow you to control every aspect of the program. PageStream's ARexx features allow you to write your own scripts and to record scripts while you use PageStream.

PageStream permits two kinds of macro scripts:
- internal PageStream macros
- external ARexx macros

Both kinds of macro scripts can be executed by assigning them to function keys, adding them to the Macros menu, or by executing them with the PageStream Macros or Execute ARexx Macro commands.

Internal PageStream Macros

Internal PageStream macros are made up of PageStream ARexx commands, but may not include any ARexx language variables or commands (such as IF...THEN or FOR...NEXT). They are executed internally without ARexx. This allows fast execution and does not require that ARexx be loaded, but limits you to the PageStream command set.

Internal PageStream macros can be created by recording program actions with the Start Recording command, or by writing a macro with the PageStream Macros command. Internal macros are stored in the PageStream3.macros file and can only be edited from within PageStream.

External ARexx Macros

External ARexx macros are made up of PageStream ARexx commands and ARexx language commands. They are executed externally with ARexx. This gives you the full power of the ARexx language and all its features, but these macros are slower to execute than internal macros.

External macros can be created by writing them with a text editor, or by creating an internal PageStream macro and converting it to an external ARexx macro. External ARexx macros are stored in separate text files and must follow all ARexx conventions.

Why Two Macro Systems?

If you are already familiar with ARexx, PageStream's ARexx implementation will be familiar to you. It follows all established ARexx guidelines and provides the standard ARexx features. ARexx requires time to learn and scripts take time to write, however, so some users may wish for a simpler system that does not require programming knowledge. PageStream's internal macros can be recorded without writing a script, and can be executed faster because they are not sent through the ARexx interpreter.

This dual macro system provides the utmost in flexibility. The commands for the two systems are the same, and are generally referred to as just 'ARexx commands'. Because you can convert an internal macro to an external ARexx macro, you can have the best of both worlds.

See Also: ARexx Command Index

1.2 Recording an Internal PageStream Macro

Recording an Internal PageStream Macro

You can record any PageStream command to create a macro which you can execute at a later time to repeat your commands. This is a good way to automate common tasks without writing an ARexx script.

To record a PageStream macro, choose the Start Recording command. This will put a 'record on' symbol (a flashing circle) in the top-right corner of the screen. Any commands you give PageStream now will be recorded,

including:
- changing tools
- choosing commands
- drawing objects
- moving, scaling objects
- key presses

Note that PageStream records commands, not actions. The act of choosing a command to open a requester is not recorded. PageStream has a non-modal macro interface. Some programs that feature recordable macros record the opening of a requester and the selection of gadgets within the requester. This is a limited type of system because it is dependent on the mode of the program when the macro script is started. PageStream does not record that the requester is opened, but simply sets the options you choose from within the requester. For example, nothing is recorded if you open a requester and close it without selecting any options.

To finish recording a PageStream macro, choose the Stop Recording command. This will remove the 'record on' symbol from the top-right corner of the screen. The Edit Macro requester will open to allow you to view, name, edit, or assign the captured PageStream macro.

Note that internal macros are not saved to the PageStream3.macros file until the Save command is selected from the Macros menu in the PageStream Macros requester.

1.3 Editing an Internal PageStream Macro

Editing an Internal PageStream Macro

Internal PageStream macros are stored in the PageStream3.macros file in the main PageStream3 directory. This file is compressed to save disk space and cannot be edited in a text editor; however, you can edit the macros directly in PageStream.

To edit a PageStream macro, choose the PageStream Macros command to open the PageStream Macros requester. This will list all the available PageStream macros. Select one from the scrolling list and choose the Edit gadget. This will open the Edit Macros requester.

Note that internal macros are not saved to the PageStream3.macros file until the Save command is selected from the Macros menu in the PageStream Macros requester.

1.4 Executing an Internal PageStream Macro

Executing an Internal PageStream Macro

Internal PageStream macros can be executed in one of four ways:
- by pressing the assigned function key combination;
- by choosing the macro from the Macros menu;
- by choosing the macro from the Macro panel; or

- by selecting the the macro from the PageStream Macros requester.

Pressing an Assigned Function Key:

You can assign PageStream macros to the 10 function keys with the PageStream Macros command. Once a macro has been assigned to a function key, you can launch it by pressing its key. Note that internal PageStream macros share the function keys with external ARexx macros.

Choosing from the Macros Menu:

You can add PageStream macros to the Macros menu with the PageStream Macros command. Once a macro name has been added to the menu, you can execute it by choosing it from the menu. Note that internal PageStream macros and external ARexx macros are displayed together in the Macros menu.

Selecting from the Macro Panel:

If the Macro panel is open (it can be opened with the Macro Panel command), you can execute a PageStream macro simply by clicking on it. Note that internal PageStream macros and external ARexx macros are displayed together in the Macro panel.

Selecting from the PageStream Macros Requester:

You can execute a PageStream macro from the PageStream Macros requester by selecting a macro from the scrolling list and choosing the Execute gadget.

1.5 Return Values and Errors

Return Values and Errors

Many of PageStream’s commands return values to the calling macro script. These are only available when the macro is an external ARexx script, because PageStream uses the standard ARexx RC and RESULT variables.

Returned Values

Returned values are always stored in the special RESULT variable, unless there is more than one returned value, in which case a stem variable is normally used. Note that in ARexx, you must enable results explicitly if you wish to receive them. This is done by including a line near the top of your ARexx macro:

```
    options results
```

Errors

It is possible for a command to fail, because the prerequisite conditions for the command did not exist. PageStream returns error codes to the special RC variable. When a command has finished executing, the RC variable can be examined to determine whether the command was successful or not. When RC is equal to zero, the command worked as expected. Any other value indicates an error.

In general, the greater the RC number, the more severe the error. There are four possible severity levels:

```
RC =  0  Normal; there was no error.
RC =  5  Slight error; not fatal.
RC = 10  Serious error.
RC = 20  Fatal Error.
```

Note: There are very few error codes in this release of PageStream.

1.6 What is ARexx?

What is ARexx?

ARexx is an interpreted programming language. Like Basic, ARexx macro scripts (programs) can be written with a text editor and then run without having to use a compiler. Unlike programming languages, however, ARexx macro scripts can combine commands that are specific to applications, as well as to ARexx. This allows an ARexx script to control PageStream.

1.7 Starting ARexx

Starting ARexx

Before you can use PageStream's ARexx features, you must start ARexx. Most users start ARexx automatically when they boot their computers by adding the ARexx program name to their User-Startup or Startup-Sequence file.

Starting ARexx Automatically

If you have not already done this, follow these steps:

1. Open a text editor.

2. Open the S:User-Startup file.

3. Add the following command at the end of the file:

   ```
   REXXMAST >NIL:
   ```

4. Save the file.

5. Reboot your Amiga.

To Start ARexx Manually from the Workbench:

1. Open the System drawer.

2. Double-click on the RexxMast icon.

To Start ARexx Manually from the Shell:

1. Open a Shell.

2. Type REXXMAST >NIL: and press Enter.

1.8 ARexx Port Address

ARexx Port Address

Each ARexx compatible program opens an ARexx port to which you can send commands. The first copy of PageStream you run will open a port called ‘PAGESTREAM’. Subsequent copies will open ports called ‘PAGESTREAM.1’, ‘PAGESTREAM.2’ etc...

To address your commands to the default PageStream ARexx port, place the ADDRESS command after the initial remark.

i.e., ADDRESS ’PAGESTREAM’

See also:

ARexx Macro Layout

1.9 Command Format

ARexx Command Reference Format

Format: provides a template for the command:

< >	Angle brackets enclose MANDATORY parameters.
[ ]	Square brackets enclose OPTIONAL parameters.
|	Vertical bars separate alternative parameters.
{ }	Parentheses enclose parameters that can be repeated as many times as required in a command. i.e., You can create multiple tabs with one settabs command.

NB: Do not type these as part of the commands. They are used only to show the command format.

Case:

The case of PageStream ARexx commands is not important. Commands and options are listed here in uppercase to distinguish them from the explanations.

Arguments and Variables:

Some commands require you to enter an argument to the command or to specify a variable to hold a result. Variables and arguments in the command descriptions are followed by a backslash and a letter to indicate their type. The following table describes each type of argument and variable:

/I Integer: Whole numbers, positive or negative. i.e., -2, -1, 0, 1, 2, 3

/D Decimal: Any number.

/P Percentage: Any number as a percent. i.e., enter 80 for 80%.

/B Boolean: 'true' or 'false'.

/A Angles: Any number between 0 and 360.

/F Filename: Used with commands that refer to a file. The filename is required for files, and the path is required for paths; the full file path may or may not be required for files, depending on the current path.

/V Variable: Used for results.

/S String: Used for text. Any character is acceptable.*

* Multiple lines in strings should be separated by a backslash and an 'n'. i.e., "This is line 1.\nThis is line 2." To enter an actual '\n', type '\\n'.

1.10 ARexx Macro Layout

ARexx Macro Layout

All ARexx Macros must begin with a comment or remark. Remarks are specified with a '/*' at the beginning and a '*/' at the end. This tells ARexx to ignore the text within the remark indicators.

After the first remark (and any other desired remarks), you must specify the PageStream ARexx port
address
so that the commands will be given to
the correct program.

Your scripts should normally enable results so that results of operations can be returned and stored in the ARexx RESULT variable.

The rest of your script must be made up of valid PageStream ARexx commands and standard ARexx functions.

ARexx Script Framework Example

```
/* SCRIPT NAME */
/* Script description */
/* Created by yourname */

ADDRESS 'PAGESTREAM'
OPTIONS RESULTS
TRACE OFF

/* MAIN PART OF SCRIPT */

etc...
```

1.11 Executing an External ARexx Macro

Executing an External ARexx Macro

External ARexx macros are normally stored in the PageStream3:Macros drawer, although you can assign the path for macros to any drawer with the Paths command.

You may wish to install your ARexx macros in the REXX: directory because then you can execute the macros from the Shell without having to type the complete path and because it will centralize your scripts. However, most people prefer to keep them in a separate directory to reduce clutter in the REXX: directory.

There are two ways to execute an external ARexx macro:

- Choose the Execute ARexx Macro command. Select a macro file from the file requester and click on Ok.

- Type ‘rx macroname’ in an AmigaDOS shell, where macroname is the name (and path) of the macro to execute. if the macro has a ‘.rexx’ file extension, you can omit the extension.

1.12 Constraints

Constraints

Objects have a constraint flag which allows their aspect ratio to be fixed so that they cannot be accidentally resized non-proportionally. When the flag is set to CONSTRAIN, they can only be resized proportionally, and when the flag is set to FREE, they can be resized non-proportionally.

1.13 Object and Article Identification numbers

Object and Article Identification numbers

Many PageStream macro commands use the OBJECTID and ARTICLEID parameters to specify which object or article is to be manipulated. ID numbers are assigned when an object or article is created. They are session-specific

```
numbers---they are not stored with the document and thus will not be the
same the next time the document is opened.

Store ID numbers when your macro creates an object or article or is
editing a selected object so that it can change it again later. The ID
numbers are not of use for anything other than macros, and the actual
number is irrelevant. Your macro should get an ID number and use it as a
handle for the object or article.
```

1.14 Document, Window, Page and Style Tag parameters

```
Document, Window, Page and Style Tag parameters

Many PageStream macro commands use the CHAPTER, WINDOW, PAGE, MASTERPAGE
and DOCUMENT parameters. Depending on which parameter is used, you may be
able to enter the document, chapter, page, masterpage or window names.
Use the following table to decide which to enter.

CHAPTER       documentname[:chaptername[:subchaptername]]
              example: 'Untitled.1:Chapter 1'
              example: 'Project.doc'
DOCUMENT      documentname
              example: 'Untitled.1'
WINDOW        [documentname~]windowname
              example: 'Untitled.1~View.1'
PAGE          [documentname[:chaptername[:subchaptername]~]pagenumber
              example: 'Untitled.1:Chapter 1~13'
PAGERANGE     [documentname[:chaptername[:subchaptername] ]pageranges
              example: 'Untitled.1:Chapter 1 4-6,7,9-12'
MASTERPAGE    [documentname[:chaptername[:subchaptername]~]masterpagename
              example: 'Untitled.1:Chapter 1~Default Master Page'
MPG           [documentname[:chaptername[:subchaptername]~]masterpagename:]side
              where side is <LEFT|CENTER|RIGHT>
              example: 'Untitled.1:Chapter 1~Default Master Page:left'
STYLETAG      [documentname[:chaptername[:subchaptername]~]styletagname
              example: 'Untitled.1~Body Text'
VARIABLE      [documentname[:chaptername[:subchaptername]~]variablename
              example: 'Untitled.1~Author'
ARTICLE       [documentname[:chaptername[:subchaptername]~]articlename
              example: 'Untitled.1~Story1'


Note that all of these parameters require string inputs except for the
PAGE parameter. If just the page number is given, you can enter the page
number without quotes. If the document name is added, you must enter it
as a string.

Note that CHAPTER can refer to a chapter or the root level of a document.
```

1.15 Converting a PageStream Macro to an ARexx Macro

Converting a PageStream Macro to an ARexx Macro

Internal PageStream macros can be converted to external ARexx macros by exporting them from the PageStream Macros requester. Select a macro script from the scrolling list and choose the Export gadget. This will save the macro as an ASCII text file.

1.16 Object Coordinates

Object Coordinates

Most objects have four coordinate values: px1, py1, px2, py2. These are the coordinates of the top/left and bottom/right corners, respectively.

Other objects have eight coordinate values: px1, py1, px2, py2, px3, py3, px4, py4. These are the coordinates of the top/left, bottom/left, bottom/right and top/right coordinates, respectively.

Some objects have a center and radii coordinate values instead of corner coordinates: cx, cy, rx, ry. These are the coordinates of the center point and horizontal and vertical radii, respectively.

1.17 Using Requesters in an ARexx Macro

Using Requesters in an ARexx Macro

ARexx has very limited facilities for user interaction. It cannot by itself open a requester from which the user can select an option or enter data.

PageStream offers powerful ARexx requester abilities through its application library engine. From ARexx, you can have many of the same abilities that the PageStream programmers have. With this privilege comes a responsability to use proper programming techniques. The requester commands are not simple to use, and if you don’t write good code, you can quickly crash PageStream.

There are several limitations on requesters used from ARexx:

- you are limited in the number and types of gadgets available. Things such as the feedback gadget in the Rotate requester are not available from ARexx, nor are icons.

- they are not asynchronous. This means that the requester must be closed for the macro to get feedback. The script will not know that the user has selected an item from a list gadget until the user clicks on an exit gadget. This rules out floating palettes and interactive requesters.

- they are slower than the normal program requesters due to technical reasons.

The two rules you must follow when using requesters from ARexx are:

- if you allocate a requester, you must deallocate it when you're done. ARexx is a forgiving language which closes files if you leave them open to clean up after messy users, but PageStream's requester commands are not as forgiving.

- if you allocate a list, you must deallocate it when you're done.

The basic structure of a script using a requester is:

```
   allocarexxlist       Allocates a list.
   addarexxlist         Adds an item to a list.
   allocarexxrequester  Allocates a requester.
   addarexxgadget       Adds a gadget.
  .
  .
   setarexxgadget       Sets a gadget.
  .
  .
   doarexxrequester     Opens a requester.
   getarexxgadget       Gets the final status of a gadget.
   freearexxrequester   Unallocates a requester.
   freearexxlist        Unallocates an arexx list.
```

If you plan to use a list gadget (popup, cycle, scrolling list), you must allocate a list for it and free it when you're done. The lists are not part of the requesters and thus can be allocated or freed at any part in the script, or used in more than one requester.

When you allocate a list or requester, PageStream will return a handle. You must use this handle to reference the list or requester for the rest of the script.

After allocating a requester, you can design it with the addarexxgadget command. Lists can be attached to list gadgets with the setarexxgadget command.

When you are done, you can display it with the doarexxrequester command. After the user has chosen an exit gadget, its handle will be returned as a result. You can then query the state of any of the non-exit gadgets in the requester with the getarexxgadget command until the requester is freed.

If you need to simulate an interactive requester from a macro, you can close the requester, check the gadget status, and then open it again.

Refer to the FontSpec.rexx macro for simple and complex examples of using requesters from ARexx.

Alert Requesters

PageStream3 provides a simple requester with the getchoice command which allows you to display a string and up to three exit gadgets; however, this requester has a fixed width and long strings will be truncated.

Here is an example of a function you can use in your own ARexx scripts that will display a requester with one exit gadget. The requester's width will change to match the length of the string you use, up to the width of the screen.

To display the alert requester, use:

```
CALL DOALERT("The sky is falling!")
```

You must include this function somewhere in your script:

```
DOALERT:
parse arg sAlertText
iAlertLength=length(sAlertText)
/* Display an alert requester */
        allocarexxrequester '"Macro Alert"' iAlertLength*8+24 55
            hAlertReq=result
        addarexxgadget hAlertReq TEXT 8 12 iAlertLength*8+8 border none string '"' ←
           sAlertText'"'
        addarexxgadget hAlertReq EXIT iALertLength*8-58 38 70 label "_Exit"
        doarexxrequester hAlertReq
        freearexxrequester hAlertReq
RETURN
```