FinalCalc.hyper

Khalid Aldoseri

COLLABORATORS

	TITLE : FinalCalc.hyper		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Khalid Aldoseri	January 17, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FinalCalc.hyper

1.1 Scripts and Macros...

```
                  Scripts and Macros:


                 >
                 Scripts

                 >
                 What is ARexx?

                 >
                 Macros

                 >
                 The Recorder

                 >
                 Running Scripts
```

1.2 Scripts...

```
Scripts:
FinalCalc can accept text commands that direct it to do anything the user
would do via the user interface, and also do things not possible via the
user interface.

A text command can be entered directly via the keyboard, run from a cell or a
range, run from an external file, sent from another program to FinalCalc or
run via the ARexx script system.

A script consists of one or more text commands grouped into a 'script' that
defines a sequence of things that FinalCalc will do.

For details on scripts and script commands, please refer to your FinalCalc
manual.
```

1.3 What is ARexx...

What is ARexx?
ARexx is a system script lanuage that comes with the Amiga Operating System,
(or purchased separately for systems running Amiga OS’s without ARexx).

As far as FinalCalc is concerned, ARexx does two main jobs:

a. It talks to programs:

ARexx can talk to any program that supports it via an ’ARexx Port’. This is a door that a program opens to the outside world, allowing commands to come through it from external programs via the ARexx system.

FinalCalc opens a port called ’FCALC.1’. Any ARexx script can send text commands to that port and FinalCalc will apply those commands to the current project.

If more than one copy of FinalCalc is currently running in the system, the second copy opens a port called ’FCALC.2’, the third copy ’FCALC.3’ and so on. You can find out the port name of the current copy by using the ’Project - About’ menu, which brings up the ’About’ requester. It tells you the current port name.

b. It provides a language:

ARexx is also a command language. It has logic and flow control as well as variables. A script can contain loops and logic to intelligently control external programs (like FinalCalc).

1.4 Macros...

Macros:
A Macro defines a script, how it is run, and where to find it.

Once a macro is defined, it can be run either in many ways, both manually and automatically.

The Macros List:

A project can have multiple macros defined. These are collectively knows as the Macros List. To access the Macros List, use the ’Tools - Macros List’ menu to bring up the ’Macros List’ requester.

It shows all macros defined for the current project. From it, you can define a new macro, edit a macro, delete and clone macros, and run a macro.

The Macro:

The New and Edit buttons in the ’Macros List’ requester (described in 9.2.1 above) call the Edit Macro requester, which defines a macro. It contains the following settings:

```
1.  Macro name:

  This is a string (upto 20 chars) which is just used as a title for the
  macro for user identification.  It defaults to something like "Macro # 1'.

2.  Macro command:

  This controls the source of the text for the script.  It can be one of
  three things, based on what the 'Commmand Source' is set to:

  a.  Sheet Range:

    The actual contents of the macro as a range in the sheet.  This can be
    either a range specification, (e.g. a1:b10) or a range name (e.g.
    'SALES').

    The contents of the range are converted into text, then run as a script.

  b.  String Command:

    A simple string entered in the command string gadget.  Multiple
    statements can be strung together by using the ; as a statement
    separator.

  c.  External File:

    The command is a filename pointing to a file that contains the script
    to be executed.

        When FinalCalc tries to run a Macro that is attached to an external file,
        it tries to locate the specified file in the following places:

          1. The current program path.
          2. The path of the project itself.
          3. The default scripts path.
          4. The default projects path.
          5. The "Rexx:" directory.
          6. The default program path.

       These paths are searched in the above order.  FinalCalc also tests to see
       if the script name does not end in ".rexx", and if so, it will try it
       with a ".rexx" extension if it doesn't find it.

       The recommended method is to place the script file for a macro in the
       same directory as the project file itself, with a .rexx extension.


  The [?] button for the Command follows the status of the Command Source.
  If the command source is 'Sheet Range' clicking the [?] button will allow
  you to select a sheet range, if the command source if set to 'External File'
  the [?] button will bring up a file requester.

3.  Macro Execution:

  A macro can be run either via FinalCalc itself, as a simple sequencial
  script, or passed on to ARexx to parse and send back a series of commands
  to FinalCalc to execute.
```

4. Hotkey:

You can assign any keyboard key to a macro. Once you define the hotkey for a macro, just hit that key when your are in a sheet view, and the attached macro will be executed.

To define the hotkey for a macro, click on the 'Set Hotkey' key. The Edit Macro requester will then ask you to hit a key on the keyboard to set the macro to. You can use any combination modifiers (shift keys, alt keys, control key) and any keyboard key. The selected key will be shown in the requester. (Note that keys on the Numeric keypad are considered to be different from the normal keys.)

Any key can be used, including tab, return, esacpe, delete, backspace, help, space and the function keys.

The left and right shift keys are considered to be the same modifier. The left and right Alt and Amiga keys are considered to be different keys.

If you set the same hotkey for than one macro, only the first macro in the Macro List with that hotkey one will be executed when you use that key.

The 'Enable' button, if turned off, disables the hotkey.

5. Timer:

You can set a macro to execute on a defined inteval basis. To define the timer interval, click on the 'Set Timer' button in the Edit Macro requester.

Set the hours, minutes and seconds interval and click on 'Use'. Once this interval is set, the macro will automatically execute every time that interval passes.

The 'Enable' button, if turned off, disables the timer.

6. Event Trigger:

You can attach a macro to a specific event in FinalCalc. This can specify that a certain macro is to be executed after each recalc, or after the project is loaded, or before the project is saved, and so on.

The following event triggers are available:

a. After recalc.
b. After Save.
c. After Load.
d. Before recalc.
e. Before Save.
f. Before Close.

Types a, b, & c can be run via ARexx or internally. Types d, e, & f can only be run internally. The reason is that internal execution locks the user out until the script is done, allowing for scripts to be run before an action.

Warning: you can create macro that calls itself. This will make it run in an infinite loop. This can happen, for example, if the macro was set to trigger before a save, and it was using the SAVE command itself.

If more than one macro is attached to an event, only the first macro in the Macros List will be executed when that event occurs. This is to prevent several macros running at the same time.

The 'After Load' trigger can be used in conjunction with the Default Startup Project to have FinalCalc automatically load a project and run a macro.

Running a Macro:

Once a macro is defined, it can be run in one of the following ways:

a. Selecting it from the 'Macros List' requester and clicking on the 'Run' button.

b. Hitting the keyboard hotkey that the macro was attached to.

c. The event trigger a macro is set to run on happened.

d. The timer interval for the macro, if set, has lapsed.

1.5 The Recorder...

The Recorder:

The Recorder is a script generator that records user actions in sheet and graph views and requesters and generates a script file that will duplicate these actions when replayed as a script.

Recording A Script:

To record a script, use the 'Tools - Recorder - Start Recording' menu. A file requester will come up asking for a name to save the script as. (".rexx" is automatically added to the name entered.)

Once this is done, FinalCalc is now recording all your actions. A 'Recording' indicator will show up in the screen title bar.

To stop recording, use the 'Tools - Recorder - Stop Recording' menu. The script file will be closed and the Recording indicator will disappear.

You can then run the script you just saved by using the 'Tools - Run Script From - File' or by attaching it to a Macro.

If you quit FinalCalc, the Recorder will close the script file before the program quits.

Recorder Settings:

The 'Edit Recorder Settings' requester (called from the 'Tools - Recorder
Settings' menu) allows you to control how the script is generated.

It controls the following settings:

a. Absolute Cursor Position:

  If turned on, a specific CELL or RANGE command is issued before every menu
  action saved. This makes the script specific in that it will repeat the
  effect to the same cells and ranges.

b. Relative Cursor Movements:

  If turned on, cursor movements and range selections in a sheet view will be
  saved to the script.

  This makes the script work relatively to the current cell and/or range in the
  sheet view when it is run, allowing you to repeat the same effect for
  different cells or ranges.

  Absolute Cursor Position (point a) overrides the effect of Relative Cursor
  Movements. You must turn it off to get the relative movement effects.

c. Use Short Command Names:

  When turned on, FinalCalc generates the short form of a command where
  possible. (e.g. R instead of RANGE, and C instead of CELL)

d. Generate Comments:

  If turned on, Finalcalc generates comments on some commands to make the
  script easier to read.

The script is saved in a user-readable and editable format. It looks like a
normal script. You can load it into a text editor and edit it if you like.

What is recorded:

The script recorder is fairly intelligent, and attempts to generate a script
that is easy to read and edit. For details on what user actions the Recorder
watches for, refer to the FinalCalc manual.

1.6 Running Scripts...

Running Scripts:

FinalCalc allows you to run scripts from several different sources:

a. A project range:
  The script is entered in a sheet range. Simply highlight the range to run
  as a script and use the 'Tools - Run Script From - Range' menu. The range
  will be converted to an ARexx script and sent to ARexx.

b. A file:

```
  This is a pre-built script entered as a text file.  Use the 'Tools - Run
  Script From - File' menu.  A File requester will come up.  Select the file
  to run.  ARexx is instructed to run that file directly.

  Note that the file requester comes up showing the directory defined in the
  'Macros & Recorder' path in the Program Paths.

c.  Direct entry:
  This is a single line command (or multiple commands separated by semi-colons)
  entered directly from the keyboard.  Use the 'Tools - Run Script From -
  Keyboard' menu (or the Alt-X hotkey) and enter the command to run and hit
  return.

d.  A Macro:
  A Macro is a pre-defined script that can be attached to any keyboard key, or
  instructed to run at certain times.

e.  External Script:
  Any external program that talks to FinalCalc via its ARexx port.

f.  Startup Script:
  If you run FinalCalc from the shell (not Workbench) you can give it a script
  that will get run right after it starts up.

  For example, if you run FinalCalc with the following shell command:

    FinalCalc -x "load my.sheet;menu ppp"

  This loads test.sheet, and then brings up the Print Jobs requester.

g.  Icons:
  Each icon in the Icon Panel executes a script command.  You can define
  any command for an icon.  If you want to call an external script file
  from an icon, use the SCRIPT command described below.



Running Scripts using the SCRIPT command:

The SCRIPT command is recommended when you want to run an external script from
another script.

For example:

    SCRIPT "myscript"

The SCRIPT command looks for the requested external script file in the
following places on your system:

  1. The current program path.
  2. The path of the current project itself.
  3. The default scripts path.
  4. The default projects path.
  5. The "Rexx:" directory.
  6. The default program path.

These paths are searched in the above order.  FinalCalc also tests to see if
```

```
the script name does not end in ".rexx", and if so, it will try it with a
".rexx" extension if it doesn’t find it.
```