openCarac API reference manual

Jonathan Certes

May 26, 2016

Chapter 1

Welcome to the openCarac API documentation!

Copyright

This documentation

This documentation has been generated by Doxygen (C):

The documentation itself is released under the GNU Free Documentation License:

Third-party software components

- TCL/TK is distributed under the terms of the BSD license.
- Doxygen is distributed under the terms of the GNU General Public License.

- Octave is distributed under the terms of the GNU General Public License.
- LaTeX is distributed under the terms of the LaTeX project public license.
- Ngspice is distributed under the terms of the modified BSD license.
- Gnucap is distributed under the terms of the GNU General Public License.
- Xyce is distributed under the terms of the GNU General Public License and other licenses.
- SMASH is a registered trademark of Dolphin Integration.
- Linux is distributed under the terms of the GNU General Public License and other licenses.
- Windows is a registered trademark of Microsoft.

All other trademarks are property of their respective owners.

1.1 Introduction

openCarac is more than a simple program coded with TCL, it is a TCL package and comes with functions that can be called in any TCL script. All these functions form the openCarac Application Programming Interface.

Since openCarac might not be fully adapted to your needs, custom procedures can be created using the API and update the openCarac main executable in your environment. Also, openCarac API functions may automatize your tasks and be called in any other script.

1.2 How to read this documentation

This documentation describes the available functions allowing to access the openCarac settings and classes. Its organization is as follows:

- The *Module* chapter contains the API modules organized by themes (global functions, classes, simulator settings...).
- The *File* chapter lists functions that are available in openCarac API but defined in separated files.

1.3 How to use openCarac API

There are various ways to use openCarac API functions in your environment ; here is a brief description of each way.

1.3.1 Using openCarac main executable

openCarac comes with a main executable that uses the API functions, it can be called through your TCL interpreter and arguments can be given. When executing openCarac with --tcl option, a TCL function is executed at the beginning of the openCarac execution. Every openCarac API function as described in this documentation is available.

The following command describes how to print a message in openCarac log file and source a script file *myFile.tcl* calling openCarac API functions:

```
tclsh openCarac.tcl --tcl "openCarac_message {Hello World.} ; source myFile.tcl"
```

1.3.2 In openCarac custom procedures

It is possible to use custom procedures in openCarac, these procedures are defined in the file customProcedures.tcl located in the user home directory. When openCarac calls one of these procedures, every openCarac API function as described in this documentation is available.

1.3.3 In any TCL script

openCarac being a TCL package, it can be used in any TCL script as soon as you call TCL function *package require*.

But first, you must configure your TCL environment : i.e. modify the *pkgIndex.tcl* file located into one of the folders defined in the *$::tcl_pkgPath* variable. Just call the following command in your TCL interpreter to access the list of possible folders:

```
puts $::tcl_pkgPath
```

Then edit the appropriate *pkgIndex.tcl* file (see TCL manual pages for more informations) and add the following lines:

```
package ifneeded openCarac 1.0 {
source [file join $dir openCaracApi.tcl]
}
```

Where *$dir* must be replaced by the path of the folder where openCaracApi.tcl file is located.

Now that your TCL environment has been updated, you can load openCarac package by adding the following line at the beginning of your code:

```
package require openCarac
```

Then every openCarac API function as described in this documentation becomes available.

1.4 openCarac API overview

openCarac aims to be an oriented-object tool whilst requiring a minimum of dependences. Since TCL is not an oriented-object language by itself ; openCarac provides functions having objects as arguments to access their attributes.

Classes hierarchy is as represented on the following figure:

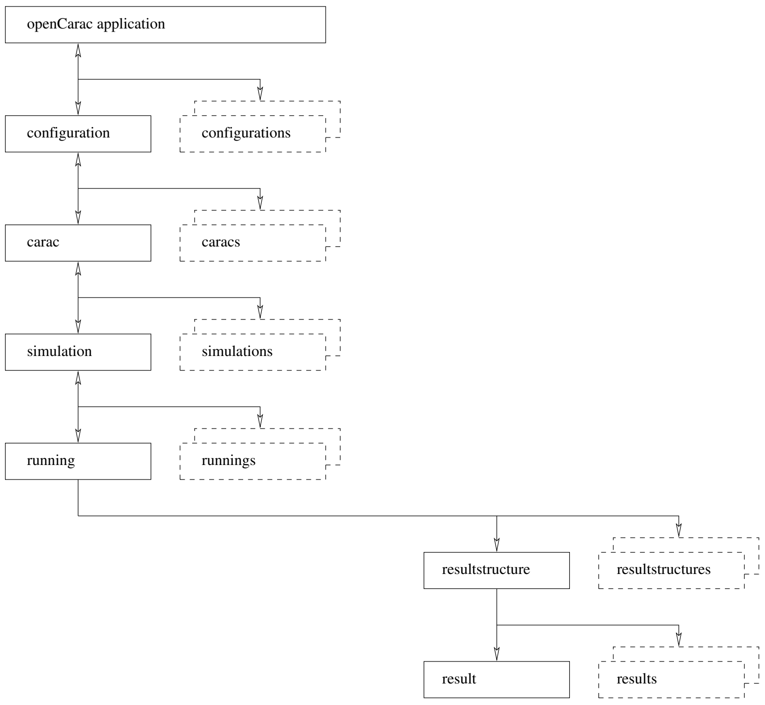

Figure 1.1: openCarac API classes diagram

- openCarac environment settings are set in an openCarac *application* namespace ; there is only one namespace.
- openCarac *configurations* do not have any parent, they can be accessed from the openCarac *application*.
- for each class in the hierarchy, the list of objects can be accessed from their parent.
- openCarac *resultstructures* and openCarac *results* do not permit to access their parents back.

More informations about functions that are specific to each class are given in the *Module* chapter.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 Global functions

Global functions of openCarac API.

Functions

- openCarac_message theString

  Prints a message in openCarac output.
- openCarac_error theString

  Prints an error in openCarac output.
- openCarac_warning theString

  Prints a warning in openCarac output.
- openCarac_exit

  Exits openCarac.
- openCarac_createTestcase

  Creates an openCarac testcase in the current directory.

4.1.1 Detailed Description

Global functions of openCarac API.

Here are defined global functions of openCarac API ; these functions do not depend on any class.

4.1.2 Function Documentation

4.1.2.1 openCarac_createTestcase

Creates an openCarac testcase in the current directory.

If you think you have discovered a bug at a given instant of the execution of openCarac, feel free to provide some feedback. This function creates a testcase folder in the current directory and copy every file loaded by openCarac when it is called. Also, a TCL script is created to replay the whole sequence. This function must be used only for debug.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# create a testcase at the end of openCarac execution:
proc openCaracHook_ON_PRE_EXIT {} {
    openCarac_createTestcase
}
```

4.1.2.2 openCarac_error *theString*

Prints an error in openCarac output.

Increments the number of errors and prints a string in the file or output selected by openCarac_applicationSetLogFile. The selected string is prefixed by "∗∗ Error ∗∗ " when printed. Number of errors can be accessed through openCarac_applicationGetNumberOfErrors.

Parameters

theString	: String.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
set myName {World}
openCarac_error "Hello $myName. An error occured."
```

4.1.2.3 openCarac_exit

Exits openCarac.

Prints a footer and then exits with the number of errors as argument. Number of errors can be accessed through openCarac_applicationGetNumberOfErrors. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_EXIT is executed.

Returns

Integer ; number of errors.

Example

```
proc main {} {
    # write your main code here
    error "my error."
}

# exit if an error occured:
if { [catch {main} returned] } {
    openCarac_error "main returned an error: $returned"
    openCarac_exit
}
```

4.1.2.4 openCarac_message *theString*

Prints a message in openCarac output.

Prints a string in the file or output selected by openCarac_applicationSetLogFile.

Parameters

theString	: String.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
set myName {World}
openCarac_message "Hello $myName."
```

4.1.2.5 openCarac_warning *theString*

Prints a warning in openCarac output.

Increments the number of warnings and prints a string in the file or output selected by openCarac_applicationSetLog↩ File. The selected string is prefixed by "∗∗ Warning ∗∗ " when printed. Number of warnings can be accessed through openCarac_applicationGetNumberOfWarnings.

Parameters

theString	: String.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
set myName {World}
openCarac_warning "Hello $myName. Be careful."
```

4.2 Application namespace

Definition of functions to interact with openCarac environement.

Functions

- openCarac_applicationGetNumberOfErrors

  Access the number of errors encountered during openCarac execution.
- openCarac_applicationGetNumberOfWarnings

  Access the number of warnings encountered during openCarac execution.
- openCarac_applicationLoadEnvironment

  Overloads the openCarac default environment settings by the user environment.
- openCarac_applicationSaveEnvironment

  Overwrites the openCarac environment files located in the user home directory to save the current settings and creates a default custom procedures file.
- openCarac_applicationRestorePreviousSession

  Add openCarac configuration file paths to the current openCarac session just as it was previously saved.
- openCarac_applicationParseArgv argv

  Calls openCarac argv parser and update openCarac current session settings.
- openCarac_applicationGetLogFile

  Returns the value of "log file" attribute of the openCarac application namespace.
- openCarac_applicationSetLogFile value

  Sets the value of "log file" attribute of openCarac application namespace.
- openCarac_applicationGetDefaultSimulator

  Returns the value of "default simulator" attribute of the openCarac application namespace.
- openCarac_applicationSetDefaultSimulator value

  Sets the value of "default simulator" attribute of openCarac application namespace.
- openCarac_applicationGetDefaultConfigurationFileName

  Returns the value of "default configuration file name" attribute of the openCarac application namespace.
- openCarac_applicationSetDefaultConfigurationFileName value

  Sets the value of "default configuration file name" attribute of openCarac application namespace.
- openCarac_applicationGetModelMarker

  Returns the value of "model marker" attribute of the openCarac application namespace.
- openCarac_applicationSetModelMarker value

  Sets the value of "model marker" attribute of openCarac application namespace.
- openCarac_applicationGetParamMarker

  Returns the value of "param marker" attribute of the openCarac application namespace.
- openCarac_applicationSetParamMarker value

  Sets the value of "param marker" attribute of openCarac application namespace.
- openCarac_applicationGetSimuMarker

  Returns the value of "simu marker" attribute of the openCarac application namespace.
- openCarac_applicationSetSimuMarker value

  Sets the value of "simu marker" attribute of openCarac application namespace.
- openCarac_applicationGetFilesExtensionFilter

  Returns the value of "files extension filter" attribute of the openCarac application namespace.
- openCarac_applicationSetFilesExtensionFilter value

  Sets the value of "files extension filter" attribute of openCarac application namespace.
- openCarac_applicationGetSimulationFileExtension

Returns the value of "simulation file extension" attribute of the openCarac application namespace.

- openCarac_applicationSetSimulationFileExtension value

  Sets the value of "simulation file extension" attribute of openCarac application namespace.
- openCarac_applicationGetNetlistFileExtension

  Returns the value of "netlist file extension" attribute of the openCarac application namespace.
- openCarac_applicationSetNetlistFileExtension value

  Sets the value of "netlist file extension" attribute of openCarac application namespace.
- openCarac_applicationActivateBatchMode

  Sets openCarac application namespace boolean "batch mode" to "1".
- openCarac_applicationDeactivateBatchMode

  Sets openCarac application namespace boolean "batch mode" to "0".
- openCarac_applicationGetBatchMode

  Returns the value of "batch mode" attribute of the openCarac application namespace.
- openCarac_applicationActivateCustomExecutionMode

  Sets openCarac application namespace boolean "custom execution mode" to "1".
- openCarac_applicationDeactivateCustomExecutionMode

  Sets openCarac application namespace boolean "custom execution mode" to "0".
- openCarac_applicationGetCustomExecutionMode

  Returns the value of "custom execution mode" attribute of the openCarac application namespace.
- openCarac_applicationActivateDebugMode

  Sets openCarac application namespace boolean "debug mode" to "1".
- openCarac_applicationDeactivateDebugMode

  Sets openCarac application namespace boolean "debug mode" to "0".
- openCarac_applicationGetDebugMode

  Returns the value of "debug mode" attribute of the openCarac application namespace.
- openCarac_applicationActivateCheckMode

  Sets openCarac application namespace boolean "check mode" to "1".
- openCarac_applicationDeactivateCheckMode

  Sets openCarac application namespace boolean "check mode" to "0".
- openCarac_applicationGetCheckMode

  Returns the value of "check mode" attribute of the openCarac application namespace.
- openCarac_applicationActivateExtractresMode

  Sets openCarac application namespace boolean "extractres mode" to "1".
- openCarac_applicationDeactivateExtractresMode

  Sets openCarac application namespace boolean "extractres mode" to "0".
- openCarac_applicationGetExtractresMode

  Returns the value of "extractres mode" attribute of the openCarac application namespace.
- openCarac_applicationActivateRunByStepMode

  Sets openCarac application namespace boolean "run by step mode" to "1".
- openCarac_applicationDeactivateRunByStepMode

  Sets openCarac application namespace boolean "run by step mode" to "0".
- openCarac_applicationGetRunByStepMode

  Returns the value of "run by step mode" attribute of the openCarac application namespace.
- openCarac_applicationActivateArchiveCreation

  Sets openCarac application namespace boolean "archive creation" to "1".
- openCarac_applicationDeactivateArchiveCreation

  Sets openCarac application namespace boolean "archive creation" to "0".
- openCarac_applicationGetArchiveCreation

  Returns the value of "archive creation" attribute of the openCarac application namespace.

- openCarac_applicationActivateFullSummaryCreation

  Sets openCarac application namespace boolean "full summary creation" to "1".
- openCarac_applicationDeactivateFullSummaryCreation

  Sets openCarac application namespace boolean "full summary creation" to "0".
- openCarac_applicationGetFullSummaryCreation

  Returns the value of "full summary creation" attribute of the openCarac application namespace.
- openCarac_applicationActivateSimulatorFilesCopy

  Sets openCarac application namespace boolean "simulator files copy" to "1".
- openCarac_applicationDeactivateSimulatorFilesCopy

  Sets openCarac application namespace boolean "simulator files copy" to "0".
- openCarac_applicationGetSimulatorFilesCopy

  Returns the value of "simulator files copy" attribute of the openCarac application namespace.
- openCarac_applicationActivateCommentOfPossibleInclusions

  Sets openCarac application namespace boolean "comment of possible inclusions" to "1".
- openCarac_applicationDeactivateCommentOfPossibleInclusions

  Sets openCarac application namespace boolean "comment of possible inclusions" to "0".
- openCarac_applicationGetCommentOfPossibleInclusions

  Returns the value of "comment of possible inclusions" attribute of the openCarac application namespace.
- openCarac_applicationActivateCreationOfHtmlFiles

  Sets openCarac application namespace boolean "creation of html files" to "1".
- openCarac_applicationDeactivateCreationOfHtmlFiles

  Sets openCarac application namespace boolean "creation of html files" to "0".
- openCarac_applicationGetCreationOfHtmlFiles

  Returns the value of "creation of html files" attribute of the openCarac application namespace.
- openCarac_applicationActivateCreationOfLatexFiles

  Sets openCarac application namespace boolean "creation of latex files" to "1".
- openCarac_applicationDeactivateCreationOfLatexFiles

  Sets openCarac application namespace boolean "creation of latex files" to "0".
- openCarac_applicationGetCreationOfLatexFiles

  Returns the value of "creation of latex files" attribute of the openCarac application namespace.
- openCarac_applicationActivateCreationOfOctaveFiles

  Sets openCarac application namespace boolean "creation of octave files" to "1".
- openCarac_applicationDeactivateCreationOfOctaveFiles

  Sets openCarac application namespace boolean "creation of octave files" to "0".
- openCarac_applicationGetCreationOfOctaveFiles

  Returns the value of "creation of octave files" attribute of the openCarac application namespace.
- openCarac_applicationPrintHeader

  Print openCarac header in the selected log file.
- openCarac_applicationPrintFooter

  Print openCarac footer in the selected log file.
- openCarac_applicationGetConfigurationFileList

  Returns the value of "configuration files list" attribute of the openCarac application namespace.
- openCarac_applicationAddConfigurationFile element

  Sets the value of "configuration files list" attribute of openCarac application namespace.
- openCarac_applicationRemoveConfigurationFile element

  Sets the value of "configuration files list" attribute of openCarac application namespace.
- openCarac_applicationGetLoadedConfigurationsList

  Returns the value of "loaded configurations list" attribute of the openCarac application namespace.
- openCarac_applicationCreateFullSummaryFile

  Creates a full check summary HTML file.

4.2.1 Detailed Description

Definition of functions to interact with openCarac environement.

Here are defined every API functions that are used to access openCarac *application* attributes. This includes the user defined environment but also what is loaded in the openCarac session.

4.2.2 Function Documentation

4.2.2.1 openCarac_applicationActivateArchiveCreation

Sets openCarac *application* namespace boolean "archive creation" to "1".

When openCarac *application* namespace boolean "archive creation" is activated, openCarac main executable proceeds to archives creation (i.e. calling openCarac_configurationCreateArchives) for each loaded openCarac *configuration* after results extraction (i.e. calling openCarac_caracExtractResults for each available openCarac *carac*). Its value can be accessed through openCarac_applicationGetArchiveCreation.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateArchiveCreation

# verify its new value:
if { [openCarac_applicationGetArchiveCreation] } {
    openCarac_message "Archive creation is activated."
} else {
    openCarac_message "Archive creation is deactivated."
}
```

4.2.2.2 openCarac_applicationActivateBatchMode

Sets openCarac *application* namespace boolean "batch mode" to "1".

When openCarac *application* namespace boolean "batch mode" is activated, openCarac main executable does not try to run graphical user interface and executes the whole sequence: for each openCarac *running* in the hierarchy of every loaded openCarac *configuration*, creates temporary folder, runs the simulator, parses the files and extracts the results. Its value can be accessed through openCarac_applicationGetBatchMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateBatchMode

# verify its new value:
if { [openCarac_applicationGetBatchMode] } {
    openCarac_message "Batch mode is activated."
} else {
    openCarac_message "Batch mode is deactivated."
}
```

4.2.2.3 openCarac_applicationActivateCheckMode

Sets openCarac *application* namespace boolean "check mode" to "1".

When openCarac *application* namespace boolean "check mode" is activated, when calling openCarac_caracMakeReady↩ForRunnings, number of openCarac *runnings* is reduced: only one openCarac *running* by simulation name and netlist combination is kept. Also, when calling openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), simulator command is executed with "check options" instead of "run options". See access functions for attributes "check options" and "run options" of the selected simulator for more informations (such as openCarac_ngspiceGetCheckOptions and openCarac_ngspiceGetRun↩Options for ngspice). Its value can be accessed through openCarac_applicationGetCheckMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateCheckMode

# verify its new value:
if { [openCarac_applicationGetCheckMode] } {
    openCarac_message "Check mode is activated."
} else {
    openCarac_message "Check mode is deactivated."
}
```

4.2.2.4 openCarac_applicationActivateCommentOfPossibleInclusions

Sets openCarac *application* namespace boolean "comment of possible inclusions" to "1".

When openCarac *application* namespace boolean "comment of possible inclusions" is activated, when creating open↩Carac *runnings* temporary folders through openCarac_runningCreateTemporaryFolder, openCarac modifies the main file to comment any model/libparam library selection or openCarac *simulation* file inclusion. A library selection is a line starting with the simulator "lib directive" and containing the tail of a possible model or libparam, i.e. a model or libparam defined in an openCarac *carac* of the same openCarac *configuration*. For more informations about simulator "lib directive", see access functions for attribute "lib directive" of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). For more informations about openCarac *carac* "model" and "libparam" attributes, see access functions openCarac_caracGet↩Model and openCarac_caracGetLibparam. An openCarac *simulation* file inclusion is a line starting with the simulator "inc directive" and containing the name of a possible openCarac *simulation*, i.e. an openCarac *simulation* defined in an open↩Carac *carac* of the same openCarac *configuration*, followed by the simulation file extension (see openCarac_application↩GetSimulationFileExtension for more informations). For more informations about simulator "inc directive", see access functions for attribute "inc directive" of the selected simulator (such as openCarac_ngspiceGetIncDirective for ngspice). For more informations about openCarac *simulation* names, see creation function openCarac_simulationGetName. When commenting in a file, the simulator "comment syntax" is added at the beginning of the line, see access functions for attribute "comment syntax" of the selected simulator (such as openCarac_ngspiceGetCommentSyntax for ngspice) for more informations. Its value can be accessed through openCarac_applicationGetCommentOfPossibleInclusions.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateCommentOfPossibleInclusions

# verify its new value:
if { [openCarac_applicationGetCommentOfPossibleInclusions] } {
    openCarac_message "Comment of possible inclusions is activated."
} else {
    openCarac_message "Comment of possible inclusions is deactivated."
}
```

4.2.2.5 openCarac_applicationActivateCreationOfHtmlFiles

Sets openCarac *application* namespace boolean "creation of html files" to "1".

When openCarac *application* namespace boolean "creation of html files" is deactivated, no HTML file is generated by openCarac. This selection affects the behaviour of results extraction and archive creation, including functions openCarac_runningExtractResults, openCarac_caracExtractResults and openCarac_configurationCreateArchives. Its value can be accessed through openCarac_applicationGetCreationOfHtmlFiles.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateCreationOfHtmlFiles

# verify its new value:
if { [openCarac_applicationGetCreationOfHtmlFiles] } {
    openCarac_message "Creation of html files is activated."
} else {
    openCarac_message "Creation of html files is deactivated."
}
```

4.2.2.6 openCarac_applicationActivateCreationOfLatexFiles

Sets openCarac *application* namespace boolean "creation of latex files" to "1".

When openCarac *application* namespace boolean "creation of latex files" is deactivated, no LaTeX file is generated by openCarac. This selection affects the behaviour of results extraction and archive creation, including functions openCarac_runningExtractResults, openCarac_caracExtractResults and openCarac_configurationCreateArchives. Its value can be accessed through openCarac_applicationGetCreationOfLatexFiles.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateCreationOfLatexFiles

# verify its new value:
if { [openCarac_applicationGetCreationOfLatexFiles] } {
    openCarac_message "Creation of latex files is activated."
} else {
    openCarac_message "Creation of latex files is deactivated."
}
```

4.2.2.7 openCarac_applicationActivateCreationOfOctaveFiles

Sets openCarac *application* namespace boolean "creation of octave files" to "1".

When openCarac *application* namespace boolean "creation of octave files" is deactivated, no GNU Octave file is generated by openCarac. This selection affects the behaviour of results extraction and archive creation, including functions openCarac_runningExtractResults, openCarac_caracExtractResults and openCarac_configurationCreateArchives. Its value can be accessed through openCarac_applicationGetCreationOfOctaveFiles.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateCreationOfOctaveFiles

# verify its new value:
if { [openCarac_applicationGetCreationOfOctaveFiles] } {
    openCarac_message "Creation of octave files is activated."
} else {
    openCarac_message "Creation of octave files is deactivated."
}
```

4.2.2.8 openCarac_applicationActivateCustomExecutionMode

Sets openCarac *application* namespace boolean "custom execution mode" to "1".

When openCarac *application* namespace boolean "custom execution mode" is activated, when calling openCarac_runningExecuteSimulator, custom procedure openCarac_customRunSimulator is called to run the simulator instead of using openCarac default behaviour. Its value can be accessed through openCarac_applicationGetCustomExecutionMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateCustomExecutionMode

# verify its new value:
if { [openCarac_applicationGetCustomExecutionMode] } {
    openCarac_message "Custom execution mode is activated."
} else {
    openCarac_message "Custom execution mode is deactivated."
}
```

4.2.2.9 openCarac_applicationActivateDebugMode

Sets openCarac *application* namespace boolean "debug mode" to "1".

When openCarac *application* namespace boolean "debug mode" is activated, when calling openCarac_runningDeleteTemporaryFolder, openCarac prints a warning and the temporary folder is not deleted. Its value can be accessed through openCarac_applicationGetDebugMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateDebugMode

# verify its new value:
if { [openCarac_applicationGetDebugMode] } {
    openCarac_message "Debug mode is activated."
} else {
    openCarac_message "Debug mode is deactivated."
}
```

4.2.2.10 openCarac_applicationActivateExtractresMode

Sets openCarac *application* namespace boolean "extractres mode" to "1".

When openCarac *application* namespace boolean "extractres mode" is activated, openCarac main executable does not create temporary folders or run the simulator: it starts with results extraction, i.e. calling openCarac_caracExtractResults for every available openCarac *carac*. Its value can be accessed through openCarac_applicationGetExtractresMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateExtractresMode

# verify its new value:
if { [openCarac_applicationGetExtractresMode] } {
    openCarac_message "Extractres mode is activated."
} else {
    openCarac_message "Extractres mode is deactivated."
}
```

4.2.2.11 openCarac_applicationActivateFullSummaryCreation

Sets openCarac *application* namespace boolean "full summary creation" to "1".

When openCarac *application* namespace boolean "full summary creation" is activated, openCarac main executable proceeds to full summary creation (i.e. calling openCarac_applicationCreateFullSummaryFile) after results extraction (i.e. calling openCarac_caracExtractResults for each available openCarac *carac*) and archives creation (i.e. calling openCarac_configurationCreateArchives for each loaded openCarac *configuration*). Its value can be accessed through openCarac_applicationGetFullSummaryCreation.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateFullSummaryCreation

# verify its new value:
if { [openCarac_applicationGetFullSummaryCreation] } {
    openCarac_message "Full summary creation is activated."
} else {
    openCarac_message "Full summary creation is deactivated."
}
```

4.2.2.12 openCarac_applicationActivateRunByStepMode

Sets openCarac *application* namespace boolean "run by step mode" to "1".

When openCarac *application* namespace boolean "run by step mode" is activated, openCarac main executable deletes temporary folders (i.e. calling openCarac_runningDeleteTemporaryFolder) for each openCarac *running* and extracts results (i.e. calling openCarac_caracExtractResults) after every simulator execution (i.e. calling openCarac_runningExecute↩ Simulator) for each openCarac *carac* instead of doing it for the whole parent openCarac *configuration*. Its value can be accessed through openCarac_applicationGetRunByStepMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateRunByStepMode

# verify its new value:
if { [openCarac_applicationGetRunByStepMode] } {
    openCarac_message "Run by step mode is activated."
} else {
    openCarac_message "Run by step mode is deactivated."
}
```

4.2.2.13 openCarac_applicationActivateSimulatorFilesCopy

Sets openCarac *application* namespace boolean "simulator files copy" to "1".

When openCarac *application* namespace boolean "simulator files copy" is activated, when parsing simulator results files of an openCarac *running* (i.e.calling openCarac_runningParseSimulatorFiles), simulator files are copied into the "open↩ CaracFiles" folder located in the openCarac *configuration* directory. The list of files to copy is filtered by extension, every file having its extension matching one the patterns defined in the "save filter" attribute of the simulator is selected for copy. See access functions for attribute "save filter" of the selected simulator for more informations (such as openCarac_ngspice↩ GetSaveFilter for ngspice). Its value can be accessed through openCarac_applicationGetSimulatorFilesCopy.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationActivateSimulatorFilesCopy

# verify its new value:
if { [openCarac_applicationGetSimulatorFilesCopy] } {
    openCarac_message "Simulator files copy is activated."
} else {
    openCarac_message "Simulator files copy is deactivated."
}
```

4.2.2.14 openCarac_applicationAddConfigurationFile *element*

Sets the value of "configuration files list" attribute of openCarac *application* namespace.

Adds an element to the list of existing openCarac *configuration* files paths. The file to add must exist and be readable. Also, an error is returned if the file path is already in the list. The full list of existing configuration files paths can be accessed through openCarac_applicationGetConfigurationFileList. This attribute is useful to know which files can be loaded with openCarac_configurationCreate or openCarac_configurationOpen. The list can be accessed through openCarac_↩ applicationGetConfigurationFileList.

Parameters

element	: String ; openCarac *configuration* file path.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
openCarac_applicationAddConfigurationFile "./a/openCarac.conf"
openCarac_applicationAddConfigurationFile "./b/openCarac.conf"

# check how many files there are:
set number [llength [openCarac_applicationGetConfigurationFileList]]
puts "There are $number configuration files to load."
```

4.2.2.15 openCarac_applicationCreateFullSummaryFile

Creates a full check summary HTML file.

In such a file can be found a tabular summarizing every results to be checked. When extracting the results on an openCarac *running*, if a measure name matches a checkmeas or a checkop of the parent openCarac *carac*, it results to the creation of an openCarac *result* that is to be checked. Every openCarac *results* having the same name will have their value printed in the same line of the tabular. Columns are separated from their files configuration: every openCarac *carac* having the same "model" and the same "libparam" are joined together and lead to the creation of a new column. This summary file is created in the current directory. The names of this file is formated using keyword *openCarac_Check_Summary* and the date and time of its creation. Results extraction is performed by openCarac_runningExtractResults. For more informations about openCarac *carac* "model" and "libparam" attributes, see access functions openCarac_caracGetModel and openCarac_caracGetLibparam. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_APPLICATION_CREATE_FULL_SUMMARY_FILE is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_APPLICATION_CREATE_FULL_SUMMARY_FILE is executed.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# extract results of every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {

    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        openCarac_caracExtractResults $theCarac
    }

}

# create a summary file:
openCarac_applicationCreateFullSummaryFile
```

4.2.2.16 openCarac_applicationDeactivateArchiveCreation

Sets openCarac *application* namespace boolean "archive creation" to "0".

When openCarac *application* namespace boolean "archive creation" is activated, openCarac main executable proceeds to archives creation (i.e. calling openCarac_configurationCreateArchives) for each loaded openCarac *configuration* after results extraction (i.e. calling openCarac_caracExtractResults for each available openCarac *carac*). Its value can be accessed through openCarac_applicationGetArchiveCreation.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateArchiveCreation

# verify its new value:
if { [openCarac_applicationGetArchiveCreation] } {
    openCarac_message "Archive creation is activated."
} else {
    openCarac_message "Archive creation is deactivated."
}
```

4.2.2.17 openCarac_applicationDeactivateBatchMode

Sets openCarac *application* namespace boolean "batch mode" to "0".

When openCarac *application* namespace boolean "batch mode" is activated, openCarac main executable does not try to run graphical user interface and executes the whole sequence: for each openCarac *running* in the hierarchy of every loaded openCarac *configuration*, creates temporary folder, runs the simulator, parses the files and extracts the results. Its value can be accessed through openCarac_applicationGetBatchMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateBatchMode

# verify its new value:
if { [openCarac_applicationGetBatchMode] } {
    openCarac_message "Batch mode is activated."
} else {
    openCarac_message "Batch mode is deactivated."
}
```

4.2.2.18 openCarac_applicationDeactivateCheckMode

Sets openCarac *application* namespace boolean "check mode" to "0".

When openCarac *application* namespace boolean "check mode" is activated, when calling openCarac_caracMakeReady↩ForRunnings, number of openCarac *runnings* is reduced: only one openCarac *running* by simulation name and netlist combination is kept. Also, when calling openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), simulator command is executed with "check options" instead of "run options". See access functions for attributes "check options" and "run options" of the selected simulator for more informations (such as openCarac_ngspiceGetCheckOptions and openCarac_ngspiceGetRun↩Options for ngspice). Its value can be accessed through openCarac_applicationGetCheckMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateCheckMode

# verify its new value:
if { [openCarac_applicationGetCheckMode] } {
    openCarac_message "Check mode is activated."
} else {
    openCarac_message "Check mode is deactivated."
}
```

4.2.2.19 openCarac_applicationDeactivateCommentOfPossibleInclusions

Sets openCarac *application* namespace boolean "comment of possible inclusions" to "0".

When openCarac *application* namespace boolean "comment of possible inclusions" is activated, when creating openCarac *runnings* temporary folders through openCarac_runningCreateTemporaryFolder, openCarac modifies the main file to comment any model/libparam library selection or openCarac *simulation* file inclusion. A library selection is a line starting with the simulator "lib directive" and containing the tail of a possible model or libparam, i.e. a model or libparam defined in an openCarac *carac* of the same openCarac *configuration*. For more informations about simulator "lib directive", see access functions for attribute "lib directive" of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). For more informations about openCarac *carac* "model" and "libparam" attributes, see access functions openCarac_caracGetModel and openCarac_caracGetLibparam. An openCarac *simulation* file inclusion is a line starting with the simulator "inc directive" and containing the name of a possible openCarac *simulation*, i.e. an openCarac *simulation* defined in an openCarac *carac* of the same openCarac *configuration*, followed by the simulation file extension (see openCarac_applicationGetSimulationFileExtension for more informations). For more informations about simulator "inc directive", see access functions for attribute "inc directive" of the selected simulator (such as openCarac_ngspiceGetIncDirective for ngspice). For more informations about openCarac *simulation* names, see creation function openCarac_simulationGetName. When commenting in a file, the simulator "comment syntax" is added at the beginning of the line, see access functions for attribute "comment syntax" of the selected simulator (such as openCarac_ngspiceGetCommentSyntax for ngspice) for more informations. Its value can be accessed through openCarac_applicationGetCommentOfPossibleInclusions.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateCommentOfPossibleInclusions

# verify its new value:
if { [openCarac_applicationGetCommentOfPossibleInclusions] } {
    openCarac_message "Comment of possible inclusions is activated."
} else {
    openCarac_message "Comment of possible inclusions is deactivated."
}
```

4.2.2.20 openCarac_applicationDeactivateCreationOfHtmlFiles

Sets openCarac *application* namespace boolean "creation of html files" to "0".

When openCarac *application* namespace boolean "creation of html files" is deactivated, no HTML file is generated by openCarac. This selection affects the behaviour of results extraction and archive creation, including functions openCarac_runningExtractResults, openCarac_caracExtractResults and openCarac_configurationCreateArchives. Its value can be accessed through openCarac_applicationGetCreationOfHtmlFiles.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateCreationOfHtmlFiles

# verify its new value:
if { [openCarac_applicationGetCreationOfHtmlFiles] } {
    openCarac_message "Creation of html files is activated."
} else {
    openCarac_message "Creation of html files is deactivated."
}
```

4.2.2.21 openCarac_applicationDeactivateCreationOfLatexFiles

Sets openCarac *application* namespace boolean "creation of latex files" to "0".

When openCarac *application* namespace boolean "creation of latex files" is deactivated, no LaTeX file is generated by openCarac. This selection affects the behaviour of results extraction and archive creation, including functions openCarac_runningExtractResults, openCarac_caracExtractResults and openCarac_configurationCreateArchives. Its value can be accessed through openCarac_applicationGetCreationOfLatexFiles.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateCreationOfLatexFiles

# verify its new value:
if { [openCarac_applicationGetCreationOfLatexFiles] } {
    openCarac_message "Creation of latex files is activated."
} else {
    openCarac_message "Creation of latex files is deactivated."
}
```

4.2.2.22 openCarac_applicationDeactivateCreationOfOctaveFiles

Sets openCarac *application* namespace boolean "creation of octave files" to "0".

When openCarac *application* namespace boolean "creation of octave files" is deactivated, no GNU Octave file is generated by openCarac. This selection affects the behaviour of results extraction and archive creation, including functions openCarac_runningExtractResults, openCarac_caracExtractResults and openCarac_configurationCreateArchives. Its value can be accessed through openCarac_applicationGetCreationOfOctaveFiles.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateCreationOfOctaveFiles

# verify its new value:
if { [openCarac_applicationGetCreationOfOctaveFiles] } {
    openCarac_message "Creation of octave files is activated."
} else {
    openCarac_message "Creation of octave files is deactivated."
}
```

4.2.2.23 openCarac_applicationDeactivateCustomExecutionMode

Sets openCarac *application* namespace boolean "custom execution mode" to "0".

When openCarac *application* namespace boolean "custom execution mode" is activated, when calling openCarac_runningExecuteSimulator, custom procedure openCarac_customRunSimulator is called to run the simulator instead of using openCarac default behaviour. Its value can be accessed through openCarac_applicationGetCustomExecutionMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateCustomExecutionMode

# verify its new value:
if { [openCarac_applicationGetCustomExecutionMode] } {
    openCarac_message "Custom execution mode is activated."
} else {
    openCarac_message "Custom execution mode is deactivated."
}
```

4.2.2.24 openCarac_applicationDeactivateDebugMode

Sets openCarac *application* namespace boolean "debug mode" to "0".

When openCarac *application* namespace boolean "debug mode" is activated, when calling openCarac_runningDeleteTemporaryFolder, openCarac prints a warning and the temporary folder is not deleted. Its value can be accessed through openCarac_applicationGetDebugMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateDebugMode

# verify its new value:
if { [openCarac_applicationGetDebugMode] } {
    openCarac_message "Debug mode is activated."
} else {
    openCarac_message "Debug mode is deactivated."
}
```

4.2.2.25 openCarac_applicationDeactivateExtractresMode

Sets openCarac *application* namespace boolean "extractres mode" to "0".

When openCarac *application* namespace boolean "extractres mode" is activated, openCarac main executable does not create temporary folders or run the simulator: it starts with results extraction, i.e. calling openCarac_caracExtractResults for every available openCarac *carac*. Its value can be accessed through openCarac_applicationGetExtractresMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateExtractresMode

# verify its new value:
if { [openCarac_applicationGetExtractresMode] } {
    openCarac_message "Extractres mode is activated."
} else {
    openCarac_message "Extractres mode is deactivated."
}
```

4.2.2.26 openCarac_applicationDeactivateFullSummaryCreation

Sets openCarac *application* namespace boolean "full summary creation" to "0".

When openCarac *application* namespace boolean "full summary creation" is activated, openCarac main executable proceeds to full summary creation (i.e. calling openCarac_applicationCreateFullSummaryFile) after results extraction (i.e. calling openCarac_caracExtractResults for each available openCarac *carac*) and archives creation (i.e. calling openCarac_configurationCreateArchives for each loaded openCarac *configuration*). Its value can be accessed through openCarac_applicationGetFullSummaryCreation.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateFullSummaryCreation

# verify its new value:
if { [openCarac_applicationGetFullSummaryCreation] } {
    openCarac_message "Full summary creation is activated."
} else {
    openCarac_message "Full summary creation is deactivated."
}
```

4.2.2.27 openCarac_applicationDeactivateRunByStepMode

Sets openCarac *application* namespace boolean "run by step mode" to "0".

When openCarac *application* namespace boolean "run by step mode" is activated, openCarac main executable deletes temporary folders (i.e. calling openCarac_runningDeleteTemporaryFolder) for each openCarac *running* and extracts results (i.e. calling openCarac_caracExtractResults) after every simulator execution (i.e. calling openCarac_runningExecuteSimulator) for each openCarac *carac* instead of doing it for the whole parent openCarac *configuration*. Its value can be accessed through openCarac_applicationGetRunByStepMode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateRunByStepMode

# verify its new value:
if { [openCarac_applicationGetRunByStepMode] } {
    openCarac_message "Run by step mode is activated."
} else {
    openCarac_message "Run by step mode is deactivated."
}
```

4.2.2.28 openCarac_applicationDeactivateSimulatorFilesCopy

Sets openCarac *application* namespace boolean "simulator files copy" to "0".

When openCarac *application* namespace boolean "simulator files copy" is activated, when parsing simulator results files of an openCarac *running* (i.e.calling openCarac_runningParseSimulatorFiles), simulator files are copied into the "openCaracFiles" folder located in the openCarac *configuration* directory. The list of files to copy is filtered by extension, every file having its extension matching one the patterns defined in the "save filter" attribute of the simulator is selected for copy. See access functions for attribute "save filter" of the selected simulator for more informations (such as openCarac_ngspiceGetSaveFilter for ngspice). Its value can be accessed through openCarac_applicationGetSimulatorFilesCopy.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_applicationDeactivateSimulatorFilesCopy

# verify its new value:
if { [openCarac_applicationGetSimulatorFilesCopy] } {
    openCarac_message "Simulator files copy is activated."
} else {
    openCarac_message "Simulator files copy is deactivated."
}
```

4.2.2.29 openCarac_applicationGetArchiveCreation

Returns the value of "archive creation" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "archive creation" is activated, openCarac main executable proceeds to archives creation (i.e. calling openCarac_configurationCreateArchives) for each loaded openCarac *configuration* after results extraction (i.e. calling openCarac_caracExtractResults for each available openCarac *carac*). Its value can be set through openCarac_applicationActivateArchiveCreation and openCarac_applicationDeactivateArchiveCreation.

Returns

Boolean ; 0 if "archive creation" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetArchiveCreation] } {
    openCarac_message "Archive creation is activated."
} else {
    openCarac_message "Archive creation is deactivated."
}
```

4.2.2.30 openCarac_applicationGetBatchMode

Returns the value of "batch mode" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "batch mode" is activated, openCarac main executable does not try to run graphical user interface and executes the whole sequence: for each openCarac *running* in the hierarchy of every loaded openCarac *configuration*, creates temporary folder, runs the simulator, parses the files and extracts the results. Its value can be set through openCarac_applicationActivateBatchMode and openCarac_applicationDeactivateBatchMode.

Returns

Boolean ; 0 if "batch mode" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetBatchMode] } {
    openCarac_message "Batch mode is activated."
} else {
    openCarac_message "Batch mode is deactivated."
}
```

4.2.2.31 openCarac_applicationGetCheckMode

Returns the value of "check mode" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "check mode" is activated, when calling openCarac_caracMakeReadyForRunnings, number of openCarac *runnings* is reduced: only one openCarac *running* by simulation name and netlist combination is kept. Also, when calling openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), simulator command is executed with "check options" instead of "run options". See access functions for attributes "check options" and "run options" of the selected simulator for more informations (such as openCarac_ngspiceGetCheckOptions and openCarac_ngspiceGetRunOptions for ngspice). Its value can be set through openCarac_applicationActivateCheckMode and openCarac_applicationDeactivateCheckMode.

Returns

Boolean ; 0 if "check mode" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetCheckMode] } {
    openCarac_message "Check mode is activated."
} else {
    openCarac_message "Check mode is deactivated."
}
```

4.2.2.32 openCarac_applicationGetCommentOfPossibleInclusions

Returns the value of "comment of possible inclusions" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "comment of possible inclusions" is activated, when creating openCarac *runnings* temporary folders through openCarac_runningCreateTemporaryFolder, openCarac modifies the main file to comment any model/libparam library selection or openCarac *simulation* file inclusion. A library selection is a line starting with the simulator "lib directive" and containing the tail of a possible model or libparam, i.e. a model or libparam defined in an openCarac *carac* of the same openCarac *configuration*. For more informations about simulator "lib directive", see access functions for attribute "lib directive" of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). For more informations about openCarac *carac* "model" and "libparam" attributes, see access functions openCarac_caracGetModel and openCarac_caracGetLibparam. An openCarac *simulation* file inclusion is a line starting with the simulator "inc directive" and containing the name of a possible openCarac *simulation*, i.e. an openCarac *simulation* defined in an openCarac *carac* of the same openCarac *configuration*, followed by the simulation file extension (see openCarac_applicationGetSimulationFileExtension for more informations). For more informations about simulator "inc directive", see access functions for attribute "inc directive" of the selected simulator (such as openCarac_ngspiceGetIncDirective for ngspice). For more informations about openCarac *simulation* names, see creation function openCarac_simulationGetName. When commenting in a file, the simulator "comment syntax" is added at the beginning of the line, see access functions for attribute "comment syntax" of the selected simulator (such as openCarac_ngspiceGetCommentSyntax for ngspice) for more informations. Its value can be set through openCarac_applicationActivateCommentOfPossibleInclusions and openCarac_applicationDeactivateCommentOfPossibleInclusions.

Returns

Boolean ; 0 if "comment of possible inclusions" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetCommentOfPossibleInclusions] } {
    openCarac_message "Comment of possible inclusions is activated."
} else {
    openCarac_message "Comment of possible inclusions is deactivated."
}
```

4.2.2.33 openCarac_applicationGetConfigurationFileList

Returns the value of "configuration files list" attribute of the openCarac *application* namespace.

Files can be added and removed with openCarac_applicationAddConfigurationFile and openCarac_applicationRemoveConfigurationFile. This attribute is useful to know which files can be loaded with openCarac_configurationCreate or openCarac_configurationOpen. Elements can be added or removed through openCarac_applicationAddConfigurationFile and openCarac_applicationRemoveConfigurationFile.

Returns

List ; strings, existing openCarac *configuration* files paths.

Example

```
openCarac_applicationAddConfigurationFile "./a/openCarac.conf"
openCarac_applicationAddConfigurationFile "./b/openCarac.conf"

# check how many files there are:
set number [llength [openCarac_applicationGetConfigurationFileList]]
puts "There are $number configuration files to load."
```

4.2.2.34 openCarac_applicationGetCreationOfHtmlFiles

Returns the value of "creation of html files" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "creation of html files" is deactivated, no HTML file is generated by openCarac. This selection affects the behaviour of results extraction and archive creation, including functions openCarac_runningExtractResults, openCarac_caracExtractResults and openCarac_configurationCreateArchives. Its value can be set through openCarac_applicationActivateCreationOfHtmlFiles and openCarac_applicationDeactivateCreationOfHtmlFiles.

Returns

Boolean ; 0 if "creation of html files" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetCreationOfHtmlFiles] } {
    openCarac_message "Creation of html files is activated."
} else {
    openCarac_message "Creation of html files is deactivated."
}
```

4.2.2.35 openCarac_applicationGetCreationOfLatexFiles

Returns the value of "creation of latex files" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "creation of latex files" is deactivated, no LaTeX file is generated by openCarac. This selection affects the behaviour of results extraction and archive creation, including functions openCarac_runningExtractResults, openCarac_caracExtractResults and openCarac_configurationCreateArchives. Its value can be set through openCarac_applicationActivateCreationOfLatexFiles and openCarac_applicationDeactivateCreationOfLatexFiles.

Returns

Boolean ; 0 if "creation of latex files" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetCreationOfLatexFiles] } {
    openCarac_message "Creation of latex files is activated."
} else {
    openCarac_message "Creation of latex files is deactivated."
}
```

4.2.2.36 openCarac_applicationGetCreationOfOctaveFiles

Returns the value of "creation of octave files" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "creation of octave files" is deactivated, no GNU Octave file is generated by openCarac. This selection affects the behaviour of results extraction and archive creation, including functions openCarac_runningExtractResults, openCarac_caracExtractResults and openCarac_configurationCreateArchives. Its value can be set through openCarac_applicationActivateCreationOfOctaveFiles and openCarac_applicationDeactivateCreationOfOctaveFiles.

Returns

Boolean ; 0 if "creation of octave files" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetCreationOfOctaveFiles] } {
    openCarac_message "Creation of octave files is activated."
} else {
    openCarac_message "Creation of octave files is deactivated."
}
```

4.2.2.37 openCarac_applicationGetCustomExecutionMode

Returns the value of "custom execution mode" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "custom execution mode" is activated, when calling openCarac_runningExecuteSimulator, custom procedure openCarac_customRunSimulator is called to run the simulator instead of using openCarac default behaviour. Its value can be set through openCarac_applicationActivateCustomExecutionMode and openCarac_applicationDeactivateCustomExecutionMode.

Returns

Boolean ; 0 if "custom execution mode" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetCustomExecutionMode] } {
    openCarac_message "Custom execution mode is activated."
} else {
    openCarac_message "Custom execution mode is deactivated."
}
```

4.2.2.38 openCarac_applicationGetDebugMode

Returns the value of "debug mode" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "debug mode" is activated, when calling openCarac_runningDeleteTemporaryFolder, openCarac prints a warning and the temporary folder is not deleted. Its value can be set through openCarac_applicationActivateDebugMode and openCarac_applicationDeactivateDebugMode.

Returns

Boolean ; 0 if "debug mode" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetDebugMode] } {
    openCarac_message "Debug mode is activated."
} else {
    openCarac_message "Debug mode is deactivated."
}
```

4.2.2.39 openCarac_applicationGetDefaultConfigurationFileName

Returns the value of "default configuration file name" attribute of the openCarac *application* namespace.

If openCarac main executable runs and the "configuration files list" attribute of openCarac *application* namespace is empty, it tries to load a file with this "default configuration file name" in the current directory. Its value cannot be an empty string or contain folder hierarchy. Its value can be set through openCarac_applicationSetDefaultConfigurationFileName.

Returns

String ; non-empty, file tail.

Example

```
# change the default configuration file name:
openCarac_applicationSetDefaultConfigurationFileName "openCarac.conf"

# check its new value:
set theConfigurationFile [openCarac_applicationGetDefaultConfigurationFileName]
puts "openCarac configuration is defined in file: $theConfigurationFile"
```

4.2.2.40 openCarac_applicationGetDefaultSimulator

Returns the value of "default simulator" attribute of the openCarac *application* namespace.

This attribute is a string in lower case corresponding to the name of the simulator that is selected by default by openCarac. When creating an openCarac *carac*, the default value for its "simulator" attribute, before being set by openCarac_caracSetSimulator, is identical to this openCarac *application* "default simulator" attribute. Its value can be set through openCarac_applicationSetDefaultSimulator.

Returns

String ; name of the simulator, in lower case.

Example

```
# get the default simulator:
set theSimulator [openCarac_applicationGetDefaultSimulator]

# set it to ngspice:
if { $theSimulator == "ngspice" } {
    openCarac_message "Default simulator already is: $theSimulator"
} else {
    openCarac_warning "Default simulator was set to: $theSimulator."
    openCarac_message "Switch it to ngspice."
    openCarac_applicationSetDefaultSimulator "ngspice"
}
```

4.2.2.41 openCarac_applicationGetExtractresMode

Returns the value of "extractres mode" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "extractres mode" is activated, openCarac main executable does not create temporary folders or run the simulator: it starts with results extraction, i.e. calling openCarac_caracExtractResults for every available openCarac *carac*. Its value can be set through openCarac_applicationActivateExtractresMode and openCarac_applicationDeactivateExtractresMode.

Returns

Boolean ; 0 if "extractres mode" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetExtractresMode] } {
    openCarac_message "Extractres mode is activated."
} else {
    openCarac_message "Extractres mode is deactivated."
}
```

4.2.2.42 openCarac_applicationGetFilesExtensionFilter

Returns the value of "files extension filter" attribute of the openCarac *application* namespace.

When creating a new or opening an existing openCarac *configuration*, openCarac parses the found files in the openCarac *configuration* folder hierarchy. Not all the files are used by openCarac: only the ones having an extension matching one of the extensions filtered by the "files extension filter" attribute. Files extension filter is a list of strings, each of them being a single word in lower case starting with a dot (.). Its value can be set through openCarac_applicationSetFilesExtension↩ Filter.

Returns

List ; strings, single words in lower case starting with a dot.

Example

```
set theExtensionsList [openCarac_applicationGetFilesExtensionFilter]

# define which files are ignored by openCarac:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*"] {

    set theExtension [string tolower [file extension $theFile]]

    if { [lsearch $theExtensionsList $theExtension] == -1 } {
        openCarac_warning "This file will be ignored by openCarac: $theFile"
    }

}
```

4.2.2.43 openCarac_applicationGetFullSummaryCreation

Returns the value of "full summary creation" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "full summary creation" is activated, openCarac main executable proceeds to full summary creation (i.e. calling openCarac_applicationCreateFullSummaryFile) after results extraction (i.e. calling openCarac_caracExtractResults for each available openCarac *carac*) and archives creation (i.e. calling open↩ Carac_configurationCreateArchives for each loaded openCarac *configuration*). Its value can be set through openCarac_↩ applicationActivateFullSummaryCreation and openCarac_applicationDeactivateFullSummaryCreation.

Returns

Boolean ; 0 if "full summary creation" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetFullSummaryCreation] } {
    openCarac_message "Full summary creation is activated."
} else {
    openCarac_message "Full summary creation is deactivated."
}
```

4.2.2.44 openCarac_applicationGetLoadedConfigurationsList

Returns the value of "loaded configurations list" attribute of the openCarac *application* namespace.

These are the openCarac *configurations* loaded in the current session. An openCarac *configuration* can be loaded with openCarac_configurationCreate or openCarac_configurationOpen.

Returns

List ; openCarac *configurations*.

Example

```
set theConfigurationsList [openCarac_applicationGetLoadedConfigurationsList]

# iterate on the configurations:
foreach theLoadedConfiguration $theConfigurationsList {

    set theFile [openCarac_configurationGetFilePath]
    openCarac_message "Loaded configuration file: $theFile"

}
```

4.2.2.45 openCarac_applicationGetLogFile

Returns the value of "log file" attribute of the openCarac *application* namespace.

This attribute may be a file path or set to one of the two values *stdout* and *stderr*. If "log file" attribute is set to a file path, then functions openCarac_message, openCarac_warning and openCarac_error print messages into the given file. If "log file" attribute is set to *stdout*, then messages are printed in the standard output. If "log file" attribute is set to *stderr*, then messages are printed in the error output. Its value can be set through openCarac_applicationSetLogFile.

Returns

String ; a file path, stdout or stderr.

Example

```
# make sure that log file is the standard output:
if { [openCarac_applicationGetLogFile] != "stdout" } {
    openCarac_applicationSetLogFile "stdout"
}
```

4.2.2.46 openCarac_applicationGetModelMarker

Returns the value of "model marker" attribute of the openCarac *application* namespace.

For openCarac to detect the main file, this model marker must be printed in it. When creating the temporary folders, model and corner substitution is performed on the line following this model marker. Model marker is a string that is not equal, without case sensitivity, to param marker (accessible through openCarac_applicationGetParamMarker) and simu marker (accessible through openCarac_applicationGetSimuMarker). Its value can be set through openCarac_applicationSet↩ModelMarker.

Returns

String.

Example

```
# set the marker value:
openCarac_applicationSetModelMarker "**myModelMarker**"

# verify its new value:
puts "openCarac param marker is now: [openCarac_applicationGetModelMarker]"
```

4.2.2.47 openCarac_applicationGetNetlistFileExtension

Returns the value of "netlist file extension" attribute of the openCarac *application* namespace.

When adding a netlist to an openCarac *carac* through openCarac_caracAddNetlist, openCarac automatically verifies that the netlist file exists. The netlist file must have an extension equal to this "netlist file extension". Netlist file extension must be a string, not a list itself, of at least two characters and starting with a dot (.). Its value can be set through openCarac_↩applicationSetNetlistFileExtension.

Returns

String ; single word, in lower case, of at least two characters and starting with a dot (.).

Example

```
# look for netlist files in the current directory:
set theExtension [openCarac_applicationGetNetlistFileExtension]
set theNetlists  [glob -nocomplain -directory [pwd] -type {f} "*$theExtension"]

openCarac_message "There are [llength $theNetlists] netlists in this folder."
```

4.2.2.48 openCarac_applicationGetNumberOfErrors

Access the number of errors encountered during openCarac execution.

Number of errors is incremented every time an error is printed by openCarac. It also is every time openCarac_error is called.

Returns

Integer ; Number of errors encountered during openCarac execution.

Example

```
puts "Number of errors is set to: [openCarac_applicationGetNumberOfErrors]"

# increment the number of errors:
openCarac_error "An error."

puts "Number of errors is set to: [openCarac_applicationGetNumberOfErrors]"
```

4.2.2.49 openCarac_applicationGetNumberOfWarnings

Access the number of warnings encountered during openCarac execution.

Number of warnings is incremented every time a warning is printed by openCarac. It also is every time openCarac_warning is called.

Returns

Integer ; Number of warnings encountered during openCarac execution.

Example

```
puts "Number of warnings is set to: [openCarac_applicationGetNumberOfWarnings]"

# increment the number of warnings:
openCarac_warning "A warning."

puts "Number of warnings is set to: [openCarac_applicationGetNumberOfWarnings]"
```

4.2.2.50 openCarac_applicationGetParamMarker

Returns the value of "param marker" attribute of the openCarac *application* namespace.

For openCarac to detect the main file, this param marker must be printed in it. When creating the temporary folders, libparam and param substitution is performed on the line following this param marker. Param marker is a string that is not equal, without case sensitivity, to model marker (accessible through openCarac_applicationGetModelMarker) and simu marker (accessible through openCarac_applicationGetSimuMarker). Its value can be set through openCarac_application↩SetParamMarker.

Returns

String.

Example

```
# set the marker value:
openCarac_applicationSetParamMarker "**myParamMarker**"

# verify its new value:
puts "openCarac param marker is now: [openCarac_applicationGetParamMarker]"
```

4.2.2.51 openCarac_applicationGetRunByStepMode

Returns the value of "run by step mode" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "run by step mode" is activated, openCarac main executable deletes temporary folders (i.e. calling openCarac_runningDeleteTemporaryFolder) for each openCarac *running* and extracts results (i.e. calling openCarac_caracExtractResults) after every simulator execution (i.e. calling openCarac_runningExecute↩ Simulator) for each openCarac *carac* instead of doing it for the whole parent openCarac *configuration*. Its value can be set through openCarac_applicationActivateRunByStepMode and openCarac_applicationDeactivateRunByStepMode.

Returns

Boolean ; 0 if "run by step mode" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetRunByStepMode] } {
    openCarac_message "Run by step mode is activated."
} else {
    openCarac_message "Run by step mode is deactivated."
}
```

4.2.2.52 openCarac_applicationGetSimulationFileExtension

Returns the value of "simulation file extension" attribute of the openCarac *application* namespace.

When adding an openCarac *simulation* to its parent openCarac *carac* through openCarac_simulationCreate, openCarac automatically verifies that the openCarac *simulation* file exists. The openCarac *simulation* file must have an extension equal to this "simulation file extension". Simulation file extension is a string, not a list itself, in lower case, of at least two characters and starting with a dot (.). Its value can be set through openCarac_applicationSetSimulationFileExtension.

Returns

String ; single word, in lower case, of at least two characters and starting with a dot (.).

Example

```
# set the simulation file extension:
openCarac_applicationSetSimulationFileExtension ".spi"

# create a simulation file:
set theBuffer [open "tran[openCarac_applicationGetSimulationFileExtension]" w]
puts  $theBuffer ".TRAN 1u 10m"
puts  $theBuffer ".PRINT TRAN V(KICK)"
puts  $theBuffer ".MEAS TRAN VPP_first_kick PP V(OUT1) FROM=1.1m TO=5.9m"
close $theBuffer
```

4.2.2.53 openCarac_applicationGetSimulatorFilesCopy

Returns the value of "simulator files copy" attribute of the openCarac *application* namespace.

When openCarac *application* namespace boolean "simulator files copy" is activated, when parsing simulator results files of an openCarac *running* (i.e.calling openCarac_runningParseSimulatorFiles), simulator files are copied into the "open↩ CaracFiles" folder located in the openCarac *configuration* directory. The list of files to copy is filtered by extension, every file having its extension matching one the patterns defined in the "save filter" attribute of the simulator is selected for copy. See access functions for attribute "save filter" of the selected simulator for more informations (such as openCarac↩ _ngspiceGetSaveFilter for ngspice). Its value can be set through openCarac_applicationActivateSimulatorFilesCopy and openCarac_applicationDeactivateSimulatorFilesCopy.

Returns

Boolean ; 0 if "simulator files copy" attribute of openCarac *application* namespace is deactivated, 1 if it is activated.

Example

```
# check the boolean value:
if { [openCarac_applicationGetSimulatorFilesCopy] } {
    openCarac_message "Simulator files copy is activated."
} else {
    openCarac_message "Simulator files copy is deactivated."
}
```

4.2.2.54 openCarac_applicationGetSimuMarker

Returns the value of "simu marker" attribute of the openCarac *application* namespace.

For openCarac to detect the main file, this simu marker must be printed in it. When creating the temporary folders, openCarac *simulation* file inclusion is substituted on the line following this simu marker. Simu marker is a string that is not equal, without case sensitivity, to model marker (accessible through openCarac_applicationGetModelMarker) and param marker (accessible through openCarac_applicationGetParamMarker). Its value can be set through openCarac_applicationSetSimuMarker.

Returns

String.

Example

```
# set the marker value:
openCarac_applicationSetSimuMarker "**mySimuMarker**"

# verify its new value:
puts "openCarac param marker is now: [openCarac_applicationGetSimuMarker]"
```

4.2.2.55 openCarac_applicationLoadEnvironment

Overloads the openCarac default environment settings by the user environment.

Parses the environment file located in the user home directory to update the settings. Also sources, in the same directory, the existing user defined custom procedures file in order to overload hook functions and make custom procedures available. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_APPLICATION_LOAD_ENVIRONMENT is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_APPLICATION_LOAD_ENVIRONMENT is executed.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# save the setting of default simulator in the environment:
openCarac_applicationSetDefaultSimulator "ngspice"
openCarac_applicationSaveEnvironment

openCarac_applicationSetDefaultSimulator "gnucap"
# load the environment:
openCarac_applicationLoadEnvironment

# check the value of default simulator:
set theSimulator [openCarac_applicationGetDefaultSimulator]
openCarac_message "Default simulator is set to: $theSimulator"
```

4.2.2.56 openCarac_applicationParseArgv *argv*

Calls openCarac argv parser and update openCarac current session settings.

According to each keyword of the *argv* (starting with "-" or "--"), updates the openCarac current session settings. If an element of the *argv* is not a keyword, it is considered as an openCarac *configuration* file path and added to the list of openCarac *configuration* files in the current session just like if it was called through openCarac_applicationAddConfigurationFile. Caution: some arguments may lead to a call of TCL exit function. A full description of the behaviour can be access by calling this procedure with "--help" argument.

Parameters

argv	: List ; arguments.

Returns

Integer ; 0 if no error occurred and if there is no need to exit, otherwise TCL exit function is called.

Example

```
# set default simulator through the API function:
openCarac_applicationSetDefaultSimulator "gnucap"
# set default simulator through the argv parser:
openCarac_applicationParseArgv "--ngspice"

# check the value of default simulator:
set theSimulator [openCarac_applicationGetDefaultSimulator]
openCarac_message "Default simulator is set to: $theSimulator"
```

4.2.2.57 openCarac_applicationPrintFooter

Print openCarac footer in the selected log file.

Print informations about the use of openCarac and the number of errors and warnings in the file or output selected by openCarac_applicationSetLogFile. Number of errors and warnings can also be accessed through openCarac_applicationGetNumberOfErrors and openCarac_applicationGetNumberOfWarnings. This has no effect on openCarac behaviour.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# print the footer in a file:
openCarac_applicationSetLogFile  "aNiceDuck.log"
openCarac_applicationPrintFooter
```

4.2.2.58 openCarac_applicationPrintHeader

Print openCarac header in the selected log file.

Print informations about the use of openCarac in the file or output selected by openCarac_applicationSetLogFile. This has no effect on openCarac behaviour.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# print the header in a file:
openCarac_applicationSetLogFile  "aNiceDuck.log"
openCarac_applicationPrintHeader
```

4.2.2.59 openCarac_applicationRemoveConfigurationFile *element*

Sets the value of "configuration files list" attribute of openCarac *application* namespace.

Removes an element from the list of existing openCarac *configuration* files paths. The element to remove to must have been added first. The full list of existing configuration files paths can be accessed through openCarac_applicationGet↵ConfigurationFileList. This attribute is useful to know which files can be loaded with openCarac_configurationCreate or openCarac_configurationOpen. The list can be accessed through openCarac_applicationGetConfigurationFileList.

Parameters

element	: String ; openCarac *configuration* file path.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
openCarac_applicationAddConfigurationFile "./a/openCarac.conf"
openCarac_applicationAddConfigurationFile "./b/openCarac.conf"

# remove one file:
openCarac_applicationRemoveConfigurationFile "./a/openCarac.conf"

# check how many files there are:
set number [llength [openCarac_applicationGetConfigurationFileList]]
puts "There are $number configuration files to load."
```

4.2.2.60 openCarac_applicationRestorePreviousSession

Add openCarac *configuration* file paths to the current openCarac session just as it was previously saved.

For each defined openCarac *configuration* file path in the previous session file in the user home directory, adds it to the list of openCarac *configuration* files in the current openCarac session. It is equivalent to calling openCarac_application↵AddConfigurationFile with every loaded openCarac *configuration* file paths the last time openCarac_applicationSave↵Environment was called.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
openCarac_applicationAddConfigurationFile "./a/openCarac.conf"
openCarac_applicationAddConfigurationFile "./b/openCarac.conf"

# saving the environment also saves the configuration files list:
openCarac_applicationSaveEnvironment

# remove the configuration files:
openCarac_applicationRemoveConfigurationFile "./a/openCarac.conf"
openCarac_applicationRemoveConfigurationFile "./b/openCarac.conf"

# reload the configuration files from the last saving:
openCarac_applicationRestorePreviousSession

# check how many files there are:
set number [llength [openCarac_applicationGetConfigurationFileList]]
puts "There are $number configuration files to load."
```

4.2.2.61 openCarac_applicationSaveEnvironment

Overwrites the openCarac environment files located in the user home directory to save the current settings and creates a default custom procedures file.

Current settings of openCarac are saved into the environment file located in the user home directory. If no custom procedures file exists, it also creates it by copying openCarac default custom procedures file. Any existing custom procedure file is not overwritten. A file containing the list of loaded openCarac *configurations* file paths is created, overwritten if it already exists, in order to restore the openCarac session through openCarac_applicationRestorePreviousSession in the future.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# save the setting of default simulator in the environment:
openCarac_applicationSetDefaultSimulator "ngspice"
openCarac_applicationSaveEnvironment

openCarac_applicationSetDefaultSimulator "gnucap"
# load the environment:
openCarac_applicationLoadEnvironment

# check the value of default simulator:
set theSimulator [openCarac_applicationGetDefaultSimulator]
openCarac_message "Default simulator is set to: $theSimulator"
```

4.2.2.62 openCarac_applicationSetDefaultConfigurationFileName *value*

Sets the value of "default configuration file name" attribute of openCarac *application* namespace.

If openCarac main executable runs and the "configuration files list" attribute of openCarac *application* namespace is empty, it tries to load a file with this "default configuration file name" in the current directory. Its value cannot be an empty string or contain folder hierarchy. Its value can be accessed through openCarac_applicationGetDefaultConfigurationFileName.

Parameters

value	: String ; non-empty, file tail.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the default configuration file name:
openCarac_applicationSetDefaultConfigurationFileName "openCarac.conf"

# check its new value:
set theConfigurationFile [openCarac_applicationGetDefaultConfigurationFileName]
puts "openCarac configuration is defined in file: $theConfigurationFile"
```

4.2.2.63 openCarac_applicationSetDefaultSimulator *value*

Sets the value of "default simulator" attribute of openCarac *application* namespace.

This attribute is a string corresponding to the name of the simulator that is selected by default by openCarac. Its value is automatically converted to lower case. When creating an openCarac *carac*, the default value for its "simulator" attribute, before being set by openCarac_caracSetSimulator, is identical to this openCarac *application* "default simulator" attribute. Its value can be accessed through openCarac_applicationGetDefaultSimulator.

Parameters

value	: String ; name of the simulator, in lower case.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# get the default simulator:
set theSimulator [openCarac_applicationGetDefaultSimulator]

# set it to ngspice:
if { $theSimulator == "ngspice" } {
    openCarac_message "Default simulator already is: $theSimulator"
} else {
    openCarac_warning "Default simulator was set to: $theSimulator."
    openCarac_message "Switch it to ngspice."
    openCarac_applicationSetDefaultSimulator "ngspice"
}
```

4.2.2.64 openCarac_applicationSetFilesExtensionFilter *value*

Sets the value of "files extension filter" attribute of openCarac *application* namespace.

When creating a new or opening an existing openCarac *configuration*, openCarac parses the found files in the openCarac *configuration* folder hierarchy. Not all the files are used by openCarac: only the ones having an extension matching one of the extensions filtered by the "files extension filter" attribute. Files extension filter must be a list of strings, each of them being a single word starting with a dot (.), openCarac automatically converts them to lower case. Its value can be accessed through openCarac_applicationGetFilesExtensionFilter.

Parameters

value	: List ; strings, single words starting with a dot.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
set theFilesExtensionsList [list]

# use the extensions of the files in the current directory:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*.*"] {

    set theExtension [string tolower [file extension $theFile]]

    if { [lsearch $theFilesExtensionsList $theExtension] == -1 } {
        lappend theFilesExtensionsList $theExtension
    }

}

# apply this filter to openCarac:
openCarac_applicationSetFilesExtensionFilter $theFilesExtensionsList
```

4.2.2.65 openCarac_applicationSetLogFile *value*

Sets the value of "log file" attribute of openCarac *application* namespace.

This attribute may be a file path or set to one of the two values *stdout* and *stderr*. If "log file" attribute is set to a file path, then functions openCarac_message, openCarac_warning and openCarac_error print messages into the given file. If "log file" attribute is set to *stdout*, then messages are printed in the standard output. If "log file" attribute is set to *stderr*, then messages are printed in the error output. Its value can be accessed through openCarac_applicationGetLogFile.

Parameters

value	: String ; a file path, stdout or stderr.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# make sure that log file is the standard output:
if { [openCarac_applicationGetLogFile] != "stdout" } {
    openCarac_applicationSetLogFile "stdout"
}
```

4.2.2.66 openCarac_applicationSetModelMarker *value*

Sets the value of "model marker" attribute of openCarac *application* namespace.

For openCarac to detect the main file, this model marker must be printed in it. When creating the temporary folders, model and corner substitution is performed on the line following this model marker. Model marker must be a string that is not equal, without case sensitivity, to param marker (accessible through openCarac_applicationGetParamMarker) and simu marker (accessible through openCarac_applicationGetSimuMarker). It should starts with a star (∗) to be a Spice comment; otherwise, a warning is printed by openCarac. Its value can be accessed through openCarac_applicationGetModelMarker.

Parameters

value	: String.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# set the marker value:
openCarac_applicationSetModelMarker "**myModelMarker**"

# verify its new value:
puts "openCarac param marker is now: [openCarac_applicationGetModelMarker]"
```

4.2.2.67 openCarac_applicationSetNetlistFileExtension *value*

Sets the value of "netlist file extension" attribute of openCarac *application* namespace.

When adding a netlist to an openCarac *carac* through openCarac_caracAddNetlist, openCarac automatically verifies that the netlist file exists. The netlist file must have an extension equal to this "netlist file extension". Netlist file extension must be a string, not a list itself, of at least two characters and starting with a dot (.). If the netlist file extension does not appear in the "files extension filter" attribute of openCarac *application* namespace, accessible through openCarac_application↩GetFilesExtensionFilter, it is automatically added. Its value can be accessed through openCarac_applicationGetNetlist↩FileExtension.

Parameters

value	: String ; single word, of at least two characters and starting with a dot (.).

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# here is a netlist file:
set theNetlistPath [file normalize "myNetlist.cir"]

# use its extension:
openCarac_applicationSetNetlistFileExtension [file extension $theNetlistPath]
```

4.2.2.68 openCarac_applicationSetParamMarker *value*

Sets the value of "param marker" attribute of openCarac *application* namespace.

For openCarac to detect the main file, this param marker must be printed in it. When creating the temporary folders, libparam and param substitution is performed on the line following this param marker. Param marker must be a string that is not equal, without case sensitivity, to model marker (accessible through openCarac_applicationGetModelMarker) and simu marker (accessible through openCarac_applicationGetSimuMarker). It should starts with a star (∗) to be a Spice comment; otherwise, a warning is printed by openCarac. Its value can be accessed through openCarac_applicationGet←ParamMarker.

Parameters

value	: String.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# set the marker value:
openCarac_applicationSetParamMarker "**myParamMarker**"

# verify its new value:
puts "openCarac param marker is now: [openCarac_applicationGetParamMarker]"
```

4.2.2.69 openCarac_applicationSetSimulationFileExtension *value*

Sets the value of "simulation file extension" attribute of openCarac *application* namespace.

When adding an openCarac *simulation* to its parent openCarac *carac* through openCarac_simulationCreate, openCarac automatically verifies that the openCarac *simulation* file exists. The openCarac *simulation* file must have an extension equal to this "simulation file extension". Simulation file extension must be a string, not a list itself, of at least two characters and starting with a dot (.). If the openCarac *simulation* file extension does not appear in the "files extension filter" attribute of openCarac *application* namespace, accessible through openCarac_applicationGetFilesExtensionFilter, it is automatically added. Its value can be accessed through openCarac_applicationGetSimulationFileExtension.

Parameters

value	: String ; single word, of at least two characters and starting with a dot (.).

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# set the simulation file extension:
openCarac_applicationSetSimulationFileExtension ".spi"

# create a simulation file:
set theBuffer [open "tran[openCarac_applicationGetSimulationFileExtension]" w]
puts  $theBuffer ".TRAN 1u 10m"
puts  $theBuffer ".PRINT TRAN V(KICK)"
puts  $theBuffer ".MEAS TRAN VPP_first_kick PP V(OUT1) FROM=1.1m TO=5.9m"
close $theBuffer
```

4.2.2.70 openCarac_applicationSetSimuMarker *value*

Sets the value of "simu marker" attribute of openCarac *application* namespace.

For openCarac to detect the main file, this simu marker must be printed in it. When creating the temporary folders, openCarac *simulation* file inclusion is substituted on the line following this simu marker. Simu marker must be a string that is not equal, without case sensitivity, to model marker (accessible through openCarac_applicationGetModelMarker) and param marker (accessible through openCarac_applicationGetParamMarker). It should starts with a star (∗) to be a Spice comment; otherwise, a warning is printed by openCarac. Its value can be accessed through openCarac_applicationGetSimuMarker.

Parameters

value	: String.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# set the marker value:
openCarac_applicationSetSimuMarker "**mySimuMarker**"

# verify its new value:
puts "openCarac param marker is now: [openCarac_applicationGetSimuMarker]"
```

4.3 Configuration class

Definition of functions to manipulate openCarac *configurations*.

Functions

- openCarac_configurationCreate theConfigurationFilePath

  Loads an empty openCarac configuration.
- openCarac_configurationOpen theConfigurationFilePath

  Loads an openCarac configuration and parses its openCarac configuration file.
- openCarac_configurationClose theConfiguration

  Unloads the openCarac configuration.
- openCarac_configurationSave theConfiguration

  Save the openCarac configuration state into its openCarac configuration file.
- openCarac_configurationGetFilePath theConfiguration

  Returns the value of "file path" attribute of the openCarac configuration.
- openCarac_configurationGetMainFilePath theConfiguration

  Returns the value of "main file path" attribute of the openCarac configuration.
- openCarac_configurationGetCaracsList theConfiguration

  Returns the value of "caracs list" attribute of the openCarac configuration.
- openCarac_configurationCreateArchives theConfiguration

  Creates archive files into the openCarac configuration directory.

4.3.1 Detailed Description

Definition of functions to manipulate openCarac *configurations*.

Here are defined every API functions that are used to access openCarac *configurations*. Various openCarac *configurations* can be loaded at the same time in openCarac. An openCarac *configuration* contains the settings of what openCarac may run the simulator with for a specific folder. Having of an openCarac *configuration* in a directory makes openCarac parse the files located in its hierarchy, identify a main file and define various openCarac *simulations* to run. Each openCarac *configuration* may have various openCarac *carac* children.

4.3.2 Function Documentation

4.3.2.1 openCarac_configurationClose *theConfiguration*

Unloads the openCarac *configuration*.

Frees the memory from the informations about the openCarac *configuration*. After calling this function, the openCarac *configuration* is not available in the list returned by openCarac_applicationGetLoadedConfigurationsList. Also, it calls openCarac_caracDelete for each openCarac *carac* children.

Parameters

theConfiguration	: openCarac *configuration*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# open the first configuration file:
set theConfFile [lindex [openCarac_applicationGetConfigurationFileList] 0]
set theConf [openCarac_configurationOpen $theConfFile]

# the carac list is not empty:
foreach theCarac [openCarac_configurationGetCaracsList $theConf] {
    set theCaracName [openCarac_caracGetName $theCarac]
    puts "The carac name is: $theCaracName"
}

openCarac_configurationClose $theConf

# now, the configuration cannot be accessed (an error code is returned):
set theGetCaracsList [openCarac_configurationGetCaracsList $theConf]
puts "The return code is: $theGetCaracsList."
```

4.3.2.2 openCarac_configurationCreate *theConfigurationFilePath*

Loads an empty openCarac *configuration*.

Loads an empty openCarac *configuration* with no openCarac *carac* child and sets its "file path" attribute. The parent directory of the openCarac *configuration* file must exist and be readable. No other openCarac *configuration* with the same file path must be loaded; "file path" attribute can be accessed through openCarac_configurationGetFilePath. When loading the openCarac *configuration*, openCarac parses every file having an extension matching one of the openCarac files extension filter (this filter can be accessed through openCarac_applicationGetFilesExtensionFilter) that is found in the same directory (files located in a "archive_∗" folder are ignored since openCarac creates them when calling openCarac_↩ configurationCreateArchives); if none of them or more than one is identified as the main file (containing each of openCarac "model marker", "param marker" and "simu marker" once), the openCarac *configuration* is not created. These markers can be accessed through openCarac_applicationGetModelMarker, openCarac_applicationGetParamMarker and openCarac↩ _applicationGetSimuMarker. The content of every file is buffered, meaning that when creating temporary folders of a child openCarac *running* in a child openCarac *carac* (i.e. when calling openCarac_runningCreateTemporaryFolder), the files are written as they were when loading the openCarac *configuration*. This is also works when creating an archive through openCarac_configurationCreateArchives.

Parameters

the↩ Configuration↩ FilePath	: String ; openCarac *configuration* file path.

Returns

Integer ; -1 if an error occurred ; otherwise, returns the openCarac *configuration*.

Example

```
# creation of an empty configuration:
set theConf [openCarac_configurationCreate "openCarac.conf"]

# the main file has been identified:
set theMainFile [openCarac_configurationGetMainFilePath $theConf]
puts "The main file is: $theMainFile"

# there is no carac:
set numberOfCaracs [llength [openCarac_configurationGetCaracsList $theConf]]
puts "The number of caracs is $numberOfCaracs."
```

4.3.2.3 openCarac_configurationCreateArchives *theConfiguration*

Creates archive files into the openCarac *configuration* directory.

Creates a folder having its name starting with "archive_" and composed of the openCarac *carac* children names and model/libparam file names. An archive is created for each couple of model/libparam files available in the openCarac *carac* children. In this folder are copied every file that has been parsed during the creation of the openCarac *configuration* ; if existing, model and libparam files are also copied into a subdirectory with the same name as their own parent directories. Also, depending on the value of openCarac *application* booleans "creation of html files", "creation of latex files" and "creation of octave files", output files are also created function of the results of every openCarac *running* in the hierarchy of openCarac *caracs* that have their model/libparam matching the archive. Values of openCarac *application* booleans can be accessed through openCarac_applicationGetCreationOfHtmlFiles., openCarac_applicationGetCreation↩ OfLatexFiles and openCarac_applicationGetCreationOfOctaveFiles. In order to properly access the results value, a results extraction must be performed before creating the archives ; this can be done through openCarac_caracExtractResults for every openCarac *caracs* or through openCarac_runningExtractResults for every openCarac *runnings* in the open↩ Carac *configuration* hierarchy. When calling this function, before executing its main code, openCarac hook openCarac↩ Hook_ON_PRE_CONFIGURATION_CREATE_ARCHIVES is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_CONFIGURATION_CREATE_ARCHIVES is executed.

Parameters

theConfiguration	: openCarac *configuration*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {

    set theConf [openCarac_configurationOpen $theConfFile]

    # results must be extracted first:
    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {
        openCarac_caracExtractResults $theCarac
    }

    # create archive folders:
    openCarac_configurationCreateArchives $theConf

}
```

4.3.2.4 openCarac_configurationGetCaracsList *theConfiguration*

Returns the value of "caracs list" attribute of the openCarac *configuration*.

These are the openCarac *carac* children. An openCarac *carac* can be loaded by parsing an openCarac *configuration* file through openCarac_configurationOpen or by creating a new one through openCarac_caracCreate.

Parameters

theConfiguration	: openCarac *configuration*.

Returns

List ; openCarac *caracs*

Example

```
# open the first configuration file:
set theConfFile [lindex [openCarac_applicationGetConfigurationFileList] 0]
set theConf [openCarac_configurationOpen $theConfFile]

# the carac list is not empty:
foreach theCarac [openCarac_configurationGetCaracsList $theConf] {
    set theCaracName [openCarac_caracGetName $theCarac]
    puts "The carac name is: $theCaracName"
}
```

4.3.2.5 openCarac_configurationGetFilePath *theConfiguration*

Returns the value of "file path" attribute of the openCarac *configuration*.

It is a normalized absolute path to access the file the openCarac *configuration* is defined in. openCarac can write this file through openCarac_configurationSave or load it through openCarac_configurationOpen.

Parameters

theConfiguration	: openCarac *configuration*.

Returns

String ; openCarac *configuration* file path ; integer -1 if an error occurred.

Example

```
# get the file path of every configuration:
foreach theConf [openCarac_applicationGetLoadedConfigurationsList] {

    set theConfFile [openCarac_configurationGetFilePath $theConf]
    # add it to the configuration file list:
    openCarac_applicationAddConfigurationFile $theConfFile

}

# save the configuration files list for a future session:
openCarac_applicationSaveEnvironment
```

4.3.2.6 openCarac_configurationGetMainFilePath *theConfiguration*

Returns the value of "main file path" attribute of the openCarac *configuration*.

It is a normalized absolute path to access the file that has been identified as the main file when creating or opening the openCarac *configuration* through openCarac_configurationCreate or openCarac_configurationOpen. This is the file that contains the markers that can be accessed through openCarac_applicationGetModelMarker, openCarac_applicationGet↩ParamMarker and openCarac_applicationGetSimuMarker.

Parameters

theConfiguration	: openCarac *configuration*.

Returns

String ; openCarac *configuration* main file path ; integer -1 if an error occurred.

Example

```
# creation of an empty configuration:
set theConf [openCarac_configurationCreate "openCarac.conf"]

# the main file has been identified:
set theMainFile [openCarac_configurationGetMainFilePath $theConf]
puts "The main file is: $theMainFile"
```

4.3.2.7 openCarac_configurationOpen *theConfigurationFilePath*

Loads an openCarac *configuration* and parses its openCarac *configuration* file.

Loads an openCarac *configuration* and sets its "file path" attribute ; then parses it to create openCarac *carac* children and their hierarchy. The openCarac *configuration* file and its parent directory must exist and be readable. No other openCarac *configuration* with the same file path must be loaded; "file path" attribute can be accessed through openCarac_configurationGetFilePath. When loading the openCarac *configuration*, openCarac parses every file having an extension matching one of the openCarac files extension filter (this filter can be accessed through openCarac_applicationGetFilesExtensionFilter) that is found in the same directory (files located in a "archive_∗" folder are ignored since openCarac creates them when calling openCarac_configurationCreateArchives); if none of them or more than one is identified as the main file (containing each of openCarac "model marker", "param marker" and "simu marker" once), the openCarac *configuration* is not created. These markers can be accessed through openCarac_applicationGetModelMarker, openCarac_applicationGetParamMarker and openCarac_applicationGetSimuMarker. The content of every file is buffered, meaning that when creating temporary folders of a child openCarac *running* in a child openCarac *carac* (i.e. when calling openCarac_runningCreateTemporaryFolder), the files are written as they were when loading the openCarac *configuration*. This is also works when creating an archive through openCarac_configurationCreateArchives. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_CONFIGURATION_OPEN is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_CONFIGURATION_OPEN is executed.

Parameters

theConfigurationFilePath	: String ; openCarac *configuration* file path.

Returns

Integer ; -1 if an error occurred ; otherwise, returns the openCarac *configuration*.

Example

```
# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {

    set theConf [openCarac_configurationOpen $theConfFile]

    # the carac list is not empty:
    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {
        set theCaracName [openCarac_caracGetName $theCarac]
        puts "The carac name is: $theCaracName"
    }

}
```

4.3.2.8 openCarac_configurationSave *theConfiguration*

Save the openCarac *configuration* state into its openCarac *configuration* file.

Writes down the openCarac *configuration* file to save the state of every openCarac *carac* child and their hierarchy ; overwrites it if it already exist. The openCarac *configuration* "file path" attribute can be accessed through openCarac_configurationGetFilePath.

Parameters

theConfiguration	: openCarac *configuration*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf [openCarac_configurationCreate "openCarac.conf"]

# create a carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# save the configuration:
openCarac_configurationSave $theConf

# re-open the file:
openCarac_configurationClose $theConf
set theConf [openCarac_configurationOpen "openCarac.conf"]

# there is one carac:
set numberOfCaracs [llength [openCarac_configurationGetCaracsList $theConf]]
puts "The number of caracs is $numberOfCaracs."
```

4.4 Carac class

Definition of functions to manipulate openCarac *caracs*.

Functions

- openCarac_caracCreate theParentConfiguration theName

  Creates an empty openCarac carac.
- openCarac_caracDelete theCarac

  Deletes an openCarac carac.
- openCarac_caracGetName theCarac

  Returns the value of "name" attribute of the openCarac carac.
- openCarac_caracSetName theCarac value

  Sets the value of "name" attribute of the openCarac carac.
- openCarac_caracGetParentConfiguration theCarac

  Returns the value of "parent openCarac configuration" attribute of the openCarac carac.
- openCarac_caracGetSimulator theCarac

  Returns the value of "simulator" attribute of the openCarac carac.
- openCarac_caracSetSimulator theCarac value

  Sets the value of "simulator" attribute of the openCarac carac.
- openCarac_caracGetModel theCarac

  Returns the value of "Model" attribute of the openCarac carac.
- openCarac_caracSetModel theCarac value

  Sets the value of "model" attribute of the openCarac carac.
- openCarac_caracGetLibparam theCarac

  Returns the value of "libparam" attribute of the openCarac carac.
- openCarac_caracSetLibparam theCarac value

  Sets the value of "libparam" attribute of the openCarac carac.
- openCarac_caracGetCustomArgs theCarac

  Returns the value of "custom args" attribute of the openCarac carac.
- openCarac_caracSetCustomArgs theCarac value

  Sets the value of "custom args" attribute of the openCarac carac.
- openCarac_caracGetCornerList theCarac

  Returns the value of "corner list" attribute of the openCarac carac.
- openCarac_caracAddCorner theCarac element

  Sets the value of "corner list" attribute of the openCarac carac.
- openCarac_caracRemoveCorner theCarac element

  Sets the value of "corner list" attribute of the openCarac carac.
- openCarac_caracGetParamList theCarac

  Returns the value of "param list" attribute of the openCarac carac.
- openCarac_caracAddParam theCarac element

  Sets the value of "param list" attribute of the openCarac carac.
- openCarac_caracRemoveParam theCarac element

  Sets the value of "param list" attribute of the openCarac carac.
- openCarac_caracGetNetlistList theCarac

  Returns the value of "netlist list" attribute of the openCarac carac.
- openCarac_caracAddNetlist theCarac element

  Sets the value of "netlist list" attribute of the openCarac carac.

- openCarac_caracRemoveNetlist theCarac element

  Sets the value of "netlist list" attribute of the openCarac carac.
- openCarac_caracGetCheckmeasList theCarac

  Returns the value of "checkmeas list" attribute of the openCarac carac.
- openCarac_caracSetCheckmeas theCarac name minValue maxValue

  Sets the value of "checkmeas list" attribute of the openCarac carac.
- openCarac_caracUnsetCheckmeas theCarac name

  Sets the value of "checkmeas list" attribute of the openCarac carac.
- openCarac_caracGetCheckopList theCarac

  Returns the value of "checkop list" attribute of the openCarac carac.
- openCarac_caracSetCheckop theCarac name minValue maxValue

  Sets the value of "checkop list" attribute of the openCarac carac.
- openCarac_caracUnsetCheckop theCarac name

  Sets the value of "checkop list" attribute of the openCarac carac.
- openCarac_caracGetExtractopFilterList theCarac

  Returns the value of "extractop filter list" attribute of the openCarac carac.
- openCarac_caracAddExtractopFilter theCarac element

  Sets the value of "extractop filter list" attribute of the openCarac carac.
- openCarac_caracRemoveExtractopFilter theCarac element

  Sets the value of "extractop filter list" attribute of the openCarac carac.
- openCarac_caracGetSimulationsList theCarac

  Returns the value of "simulations list" attribute of the openCarac carac.
- openCarac_caracMakeReadyForRunnings theCarac

  Creates every openCarac runnings, children of openCarac simulation, for the whole openCarac carac.
- openCarac_caracExtractResults theCarac

  Extract results of every openCarac runnings, children of openCarac simulation, for the whole openCarac carac.

4.4.1 Detailed Description

Definition of functions to manipulate openCarac *caracs*.

Here are defined every API functions that are used to access openCarac *caracs*. Various openCarac *caracs* can be defined per openCarac *configuration*. An openCarac *carac* contains various openCarac *simulation* definitions and specifications for a specific model/libparam couple. In each of them is defined a list of corners/params to sweep, various netlists to substitute and various openCarac *simulations*. Also, data to check or extract after the simulator execution is defined in the openCarac *carac*. Each openCarac *carac* may have various openCarac *simulation* children.

4.4.2 Function Documentation

4.4.2.1 openCarac_caracAddCorner *theCarac element*

Sets the value of "corner list" attribute of the openCarac *carac*.

Appends an element to the list of corners that are used to create openCarac *runnings* in the openCarac *carac* hierarchy. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a model file is loaded. To load the "model" file (which can be accessed through openCarac_caracGetModel), open↩Carac adds it in the main file on the line following the "model marker" (accessible through openCarac_applicationGet↩ModelMarker) after a "lib directive". The corner list is a list of strings to substitute after the model in the main file. These strings are not lists themselves and cannot be empty strings. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). The element must not be a list itself or an empty string. An element cannot be added if it already is present in the corner list (case is not sensitive). The list can be accessed through openCarac_caracGetCornerList.

Parameters

theCarac	: openCarac *carac*.
element	: String ; corner, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_caracAddCorner $theCarac "TYPICAL"

# the corner list is not empty:
set numberOfCorners [llength [openCarac_caracGetCornerList $theCarac]]
puts "The number of corners is $numberOfCorners."
```

4.4.2.2 openCarac_caracAddExtractopFilter *theCarac element*

Sets the value of "extractop filter list" attribute of the openCarac *carac*.

Adds an element to the extractop filters list of the openCarac *carac*. If the "extractop filter list" is not empty, when parsing simulator files through openCarac_runningParseSimulatorFiles, instance parameters are extracted only if they match an element the "extractop filter list". Matching follows the rules of TCL "string match" command. It must be a string that is not empty and not a list itself. The element is not added if it already available in the list. To define it, case sensitivity depends on the "case sensitivity" attribute of the selected simulator. The value of the selected simulator can be accessed through openCarac_caracGetSimulator. For more information about simulator case sensitivity, see access functions for attribute "case sensitivity" of the selected simulator (such as openCarac_ngspiceGetCaseSensitivity for ngspice). The list can be accessed through openCarac_caracGetExtractopFilterList.

Parameters

theCarac	: openCarac *carac*.
element	: String ; extractop filter, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_caracAddExtractopFilter $theCarac "XDUT.M1"

# the extractop filter list is not empty:
set numberOfExtractopFilters [llength [openCarac_caracGetExtractopFilterList $theCarac]]
puts "The number of extractop filters is $numberOfExtractopFilters."
```

4.4.2.3 openCarac_caracAddNetlist *theCarac element*

Sets the value of "netlist list" attribute of the openCarac *carac*.

Appends an element to the list of netlists that are used to create openCarac *runnings* in the openCarac *carac* hierarchy. An element cannot be added if it already is present in the netlist list (case is sensitive). In order to add a netlist, make sure that the file exists with the appropriate extension (see openCarac_applicationGetNetlistFileExtension for more informations) in the same directory as the main file ; also make sure that there is one netlist inclusion in the main file and that this inclusion does not follow a marker. For more informations about file inclusion, see access functions for attribute "inc directive" of the selected simulator (such as openCarac_ngspiceGetIncDirective for ngspice). For more informations about markers, see access functions openCarac_applicationGetModelMarker, openCarac_applicationGetParamMarker and openCarac_applicationGetSimuMarker. The list can be accessed through openCarac_caracGetNetlistList. Changing the netlist list affects the behaviour of openCarac_caracMakeReadyForRunnings.

Parameters

theCarac	: openCarac *carac*.
element	: String ; netlist, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_applicationSetNetlistFileExtension ".nsx"
# file myCircuit.nsx must exist:
openCarac_caracAddNetlist $theCarac "myCircuit"

# the netlist list is not empty:
set numberOfNetlists [llength [openCarac_caracGetNetlistList $theCarac]]
puts "The number of netlists is $numberOfNetlists."
```

4.4.2.4 openCarac_caracAddParam *theCarac element*

Sets the value of "param list" attribute of the openCarac *carac*.

Appends an element to the list of params that are used to create openCarac *runnings* in the openCarac *carac* hierarchy. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a libparam file is loaded. To load the "libparam" file (which can be accessed through openCarac_caracGetLibparam), openCarac adds it in the main file on the line following the "param marker" (accessible through openCarac_applicationGetParamMarker) after a "lib directive". The param list is a list of strings to substitute after the libparam in the main file. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). The element must not be a list itself or an empty string. An element cannot be added if it already is present in the param list (case is not sensitive). The list can be accessed through openCarac_caracGetParamList.

Parameters

theCarac	: openCarac *carac*.
element	: String ; param, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_caracAddParam $theCarac "TYPICAL"

# the param list is not empty:
set numberOfParams [llength [openCarac_caracGetParamList $theCarac]]
puts "The number of params is $numberOfParams."
```

4.4.2.5 openCarac_caracCreate *theParentConfiguration theName*

Creates an empty openCarac *carac*.

Adds this new openCarac *carac* to the hierarchy of the parent openCarac *configuration*. Every attribute of the created openCarac *carac* remains empty. When openCarac extracts results (i.e. when calling openCarac_caracExtractResults), an "openCaracFiles" folder is created in the same directory as the parent openCarac *configuration* ; the openCarac *carac* name is used as a folder name. This forces the openCarac *carac* name to be formated as a correct folder name and that each openCarac *carac* of the same openCarac *configuration* has a unique name. List of available openCarac *caracs* in the parent openCarac *configuration* can be accessed through openCarac_configurationGetCaracsList. Also, spaces are not allowed in the openCarac *carac* name.

Parameters

theParent↔ Configuration	: openCarac *configuration*. This openCarac *configuration* must exist and be loaded in openCarac.
theName	: String ; correct folder name with no space character.

Returns

Integer ; -1 if an error occurred ; otherwise, returns the openCarac *carac*.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# the carac is available:
puts "The carac name is: [openCarac_caracGetName $theCarac]"
```

4.4.2.6 openCarac_caracDelete *theCarac*

Deletes an openCarac *carac*.

Frees the memory from the informations about the openCarac *carac*. After calling this function, the openCarac *carac* is not available in the list returned by openCarac_configurationGetCaracsList and every openCarac *simulation* children is not available.

Parameters

theCarac	: openCarac *carac*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# deletion of the carac:
openCarac_caracDelete $theCarac

# now, the carac cannot be accessed (an error code is returned):
set theSimu  [openCarac_simulationCreate $theCarac "transient"]
puts "The return code is: $theSimu."
```

4.4.2.7 openCarac_caracExtractResults *theCarac*

Extract results of every openCarac *runnings*, children of openCarac *simulation*, for the whole openCarac *carac*.

Depending on the values of openCarac *application* booleans "creation of html files", "creation of latex files" and "creation of octave files", creates output files for each openCarac *running*. This has the same effect as calling openCarac_runningExtractResults for each of them. Also creates HTML index and summary files, and LaTeX index and summary files. Files are created in a "openCaracFiles" folder located next to the parent openCarac *configuration* file. Parent openCarac *configuration* can be accessed through openCarac_caracGetParentConfiguration and openCarac *configuration* file path through openCarac_configurationGetFilePath. Note that, first, measures must have been added to openCarac *runnings* (through openCarac_runningSetMeasure) and results must have been saved (through openCarac_runningSaveResults). Parsing simulator files in the temporary folder (with openCarac_runningParseSimulatorFiles) also does these two steps. When creating output files, openCarac makes special treatment for any result to be checked, i.e. any measure having its name matching a checkmeas name or matching a string starting with "V" and matching a checkop name between parentheses. Indeed, in output files, results values are checked function of checkmeas or checkop values. Matching follows the rules of TCL "string match" command. Both checkmeas list and checkop list can be accessed through openCarac_caracGetCheckmeasList and openCarac_caracGetCheckopList. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_CARAC_EXTRACT_RESULTS is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_CARAC_EXTRACT_RESULTS is executed.

Parameters

theCarac	: openCarac *carac*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {

    set theConf [openCarac_configurationOpen $theConfFile]

    # extraction of the results:
    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {
        openCarac_caracExtractResults $theCarac
    }

    # create archive folders:
    openCarac_configurationCreateArchives $theConf

}
```

4.4.2.8 openCarac_caracGetCheckmeasList *theCarac*

Returns the value of "checkmeas list" attribute of the openCarac *carac*.

Every checkmeas element is a list itself, composed of three elements: the first one is its name, the second one is its minimum value and the third one is its maximum value. Its name is a string that is not empty and not a list itself. Its minimum value and maximum values are both doubles, both different from "NaN" ; minimum value is always inferior or equal to maximum value. Checkmeases can be set or unset through openCarac_caracSetCheckmeas and open↵Carac_caracUnsetCheckmeas. Changing the checkmeas list affects the behaviour of openCarac_caracExtractResults and openCarac_runningExtractResults. It also has an impact on openCarac_configurationCreateArchives and open↵Carac_applicationCreateFullSummaryFile. Elements can be added or removed through openCarac_caracSetCheckmeas and openCarac_caracUnsetCheckmeas.

Parameters

theCarac	: openCarac *carac*.

Returns

List ; lists of three elements, a string and two doubles.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# checkmeas definition:
openCarac_caracSetCheckmeas $theCarac "PHASE_MARGIN" 60 180
openCarac_caracSetCheckmeas $theCarac "DC_GAIN"      42 Inf

# access their values:
foreach theCheckmeas [openCarac_caracGetCheckmeasList $theCarac] {
    set theName     [lindex $theCheckmeas 0]
    set theMinValue [lindex $theCheckmeas 1]
    set theMaxValue [lindex $theCheckmeas 2]

    puts "specification: $theMinValue < $theName < $theMaxValue"
}
```

4.4.2.9 openCarac_caracGetCheckopList *theCarac*

Returns the value of "checkop list" attribute of the openCarac *carac*.

Every checkop element is a list itself, composed of three elements: the first one is its name, the second one is its minimum value and the third one is its maximum value. Its name is a string that is not empty and not a list itself. Its minimum value and maximum values are both doubles, both different from "NaN" ; minimum value is always inferior or equal to maximum value. Checkops can be set or unset through openCarac_caracSetCheckop and openCarac_caracUnsetCheckop. Changing the checkop list affects the behaviour of openCarac_runningParseSimulatorFiles, openCarac_caracExtractResults and openCarac_runningExtractResults. It also has an impact on openCarac_configurationCreateArchives and openCarac_↵applicationCreateFullSummaryFile. Elements can be added or removed through openCarac_caracSetCheckop and open↵Carac_caracUnsetCheckop.

Parameters

theCarac	: openCarac *carac*.

Returns

List ; lists of three elements, a string and two doubles.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# checkop definition:
openCarac_caracSetCheckop $theCarac "OUTPUT" 3   3.3
openCarac_caracSetCheckop $theCarac "INPUT"  1.5 1.8

# access their values:
foreach theCheckop [openCarac_caracGetCheckopList $theCarac] {
    set theName     [lindex $theCheckop 0]
    set theMinValue [lindex $theCheckop 1]
    set theMaxValue [lindex $theCheckop 2]

    puts "specification: $theMinValue < V($theName) < $theMaxValue"
}
```

4.4.2.10 openCarac_caracGetCornerList *theCarac*

Returns the value of "corner list" attribute of the openCarac *carac*.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a model file is loaded. To load the "model" file (which can be accessed through openCarac_caracGetModel), openCarac adds it in the main file on the line following the "model marker" (accessible through openCarac_applicationGetModelMarker) after a "lib directive". The corner list is a list of strings to substitute after the model in the main file. These strings are not lists themselves and cannot be empty strings. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). Elements can be added or removed through openCarac_caracAddCorner and openCarac_caracRemoveCorner.

Parameters

theCarac	: openCarac *carac*.

Returns

List ; Strings that are not empty and not lists themselves.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_caracAddCorner $theCarac "TYPICAL"
openCarac_caracAddCorner $theCarac "WORST"

# the corner list is not empty:
foreach theCorner [openCarac_caracGetCornerList $theCarac] {
    puts "This is a corner: $theCorner."
}
```

4.4.2.11 openCarac_caracGetCustomArgs *theCarac*

Returns the value of "custom args" attribute of the openCarac *carac*.

Custom args remain unused by openCarac in most situations but it can be accessed by the user in a hook or a custom procedure (such as openCaracHook_ON_POST_RUNNING_EXECUTE_SIMULATOR or openCarac_customRunSimulator) to change a behaviour function of an openCarac *carac*. Its value can be set through openCarac_caracSetCustomArgs.

Parameters

theCarac	: openCarac *carac*.

Returns

String ; custom args.

Example

```
# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {

    set theConf [openCarac_configurationOpen $theConfFile]

    # the carac list is not empty:
    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {
        set theCustomArgs [openCarac_caracGetCustomArgs $theCarac]
        puts "The carac custom arguments are: $theCustomArgs"
    }

}
```

4.4.2.12 openCarac_caracGetExtractopFilterList *theCarac*

Returns the value of "extractop filter list" attribute of the openCarac *carac*.

Every extractop filter is a string that is not empty and not a list itself. If the "extractop filter list" is not empty, when parsing simulator files through openCarac_runningParseSimulatorFiles, instance parameters are extracted only if they match an element the "extractop filter list". Matching follows the rules of TCL "string match" command. Extractop filters can be added or removed through openCarac_caracAddExtractopFilter and openCarac_caracRemoveExtractopFilter. Elements can be added or removed through openCarac_caracAddExtractopFilter and openCarac_caracRemoveExtractopFilter.

Parameters

theCarac	: openCarac *carac*.

Returns

List ; strings, not empty and not lists themselves.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_caracAddExtractopFilter $theCarac "XDUT.M1"
openCarac_caracAddExtractopFilter $theCarac "XDUT.M2"

# the extractop filter list is not empty:
foreach theExtractopFilter [openCarac_caracGetExtractopFilterList $theCarac] {
    puts "This is a extractop filter: $theExtractopFilter."
}
```

4.4.2.13 openCarac_caracGetLibparam *theCarac*

Returns the value of "libparam" attribute of the openCarac *carac*.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a libparam file is loaded. To load the "libparam" file, openCarac adds it in the main file on the line following the "param marker" (accessible through openCarac_applicationGetParamMarker) after a "lib directive". It also adds a "param" at the end of the "libparam" file loading. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). For more informations about "param" attribute, see access function openCarac_caracGetParamList. The libparam is either an empty string (no libparam to substitute) or a path to access the existing libparam file. This path can be absolute or relative to the parent openCarac *configuration* file. Configuration file path can be accessed through openCarac_configurationGetFilePath. Its value can be set through openCarac_caracSetLibparam.

Parameters

theCarac	: openCarac *carac*.

Returns

String ; libparam file path.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# the libparam file must exist:
openCarac_caracSetLibparam $theCarac "../../parameters/conditions.lib"

set theLibparam [openCarac_caracGetLibparam $theCarac]
puts "The libparam is now set to: $theLibparam"
```

4.4.2.14 openCarac_caracGetModel *theCarac*

Returns the value of "Model" attribute of the openCarac *carac*.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a model file is loaded. To load the "model" file, openCarac adds it in the main file on the line following the "model marker" (accessible through openCarac_applicationGetModelMarker) after a "lib directive". It also adds a "corner" at the end of the "model" file loading. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). For more informations about "corner" attribute, see access function openCarac_caracGetCornerList. The model is either an empty string (no model to substitute) or a path to access the existing model file. This path can be absolute or relative to the parent openCarac *configuration* file. Configuration file path can be accessed through openCarac_configurationGetFilePath. Its value can be set through openCarac_caracSet↩
Model.

Parameters

theCarac	: openCarac *carac*.

Returns

String ; model file path.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# the model file must exist:
openCarac_caracSetModel $theCarac "../../foundryModels/models.lib"

set theModel [openCarac_caracGetModel $theCarac]
puts "The model is now set to: $theModel"
```

4.4.2.15 openCarac_caracGetName *theCarac*

Returns the value of "name" attribute of the openCarac *carac*.

Its value is a string that is not empty and not a list itself. It is formated to be a file name. It is not possible to have various openCarac *caracs* with the same name in an openCarac *configuration*. Its value can be set in an openCarac *configuration* file, when creating an openCarac *carac* through openCarac_caracCreate or by calling openCarac_caracSetName.

Parameters

theCarac	: openCarac *carac*.

Returns

String ; openCarac *carac* name.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# change the carac name:
openCarac_caracSetName $theCarac "aBrandNewName"

# access it:
puts "The carac name is: [openCarac_caracGetName $theCarac]"
```

4.4.2.16 openCarac_caracGetNetlistList *theCarac*

Returns the value of "netlist list" attribute of the openCarac *carac*.

Changing the netlist list affects the behaviour of openCarac_caracMakeReadyForRunnings. The netlist list is a list of strings to substitute a file inclusion, with netlist extension (see openCarac_applicationGetNetlistFileExtension for more informations), in the main file when creating a temporary folder (i.e. through openCarac_runningCreateTemporaryFolder). For more informations about file inclusion, see access functions for attribute "inc directive" of the selected simulator (such as openCarac_ngspiceGetIncDirective for ngspice). These strings cannot be empty strings, they are tails of file root names (with no extension and no hierarchy) representing existing files in the openCarac *configuration* directory. Elements can be added or removed through openCarac_caracAddNetlist and openCarac_caracRemoveNetlist.

Parameters

theCarac	: openCarac *carac*.

Returns

List ; Strings that are not empty and not lists themselves.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_applicationSetNetlistFileExtension ".nsx"
# file myCircuit.nsx must exist:
openCarac_caracAddNetlist $theCarac "myCircuit"
# file myOtherCircuit.nsx must exist:
openCarac_caracAddNetlist $theCarac "myOtherCircuit"

# the netlist list is not empty:
foreach theNetlist [openCarac_caracGetNetlistList $theCarac] {
    puts "This is a netlist: $theNetlist."
}
```

4.4.2.17 openCarac_caracGetParamList *theCarac*

Returns the value of "param list" attribute of the openCarac *carac*.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a libparam file is loaded. To load the "libparam" file (which can be accessed through openCarac_caracGetLibparam), openCarac adds it in the main file on the line following the "param marker" (accessible through openCarac_application↵GetParamMarker) after a "lib directive". The param list is a list of strings to substitute after the libparam in the main file. These strings are not lists themselves and cannot be empty strings. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). Elements can be added or removed through openCarac_caracAddParam and openCarac_caracRemoveParam.

Parameters

theCarac	: openCarac *carac*.

Returns

List ; Strings that are not empty and not lists themselves.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_caracAddParam $theCarac "TYPICAL"
openCarac_caracAddParam $theCarac "WORST"

# the param list is not empty:
foreach theParam [openCarac_caracGetParamList $theCarac] {
    puts "This is a param: $theParam."
}
```

4.4.2.18 openCarac_caracGetParentConfiguration *theCarac*

Returns the value of "parent openCarac *configuration"* attribute of the openCarac *carac*.

Parameters

theCarac	: openCarac *carac*.

Returns

openCarac *configuration* ; integer -1 if an error occurred.

Example

```
# creation of a list of caracs:
set theCaracList [list]

# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {
    set theConf [openCarac_configurationOpen $theConfFile]
    # access their caracs:
    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {
        lappend theCaracList $theCarac
    }
}

# get the configuration back from the carac:
foreach theCarac $theCaracList {
    set theConf     [openCarac_caracGetParentConfiguration $theCarac]
    set theConfFile [openCarac_configurationGetFilePath $theConf]
    puts "The carac is defined in file: $theConfFile"
}
```

4.4.2.19 openCarac_caracGetSimulationsList *theCarac*

Returns the value of "simulations list" attribute of the openCarac *carac*.

These are the openCarac *simulation* children. An openCarac *simulation* can be loaded by parsing an openCarac *configuration* file (through openCarac_configurationOpen) or by creating a new one (through openCarac_simulationCreate).

Parameters

theCarac	: openCarac *carac*.

Returns

List ; openCarac *simulations* ; integer -1 if an error occurred.

Example

```
# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {

    set theConf [openCarac_configurationOpen $theConfFile]

    # access the caracs:
    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {
        # access the simulations:
        foreach theSimulation [openCarac_caracGetSimulationsList $theCarac] {
            # print their file path:
            set theSimuFile [openCarac_simulationGetFilePath $theSimulation]
            puts "This is a simulation file: $theSimuFile"
        }
    }

}
```

4.4.2.20 openCarac_caracGetSimulator *theCarac*

Returns the value of "simulator" attribute of the openCarac *carac*.

Selecting a simulator affects the behaviour of openCarac_runningExecuteSimulator and openCarac_runningParseSimulatorFiles. Its value can be set through openCarac_caracSetSimulator.

Parameters

theCarac	: openCarac *carac*.

Returns

String ; name of the simulator, in lower case.

Example

```
# get the default simulator:
set theDefaultSimulator [openCarac_applicationGetDefaultSimulator]

# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {

    set theConf [openCarac_configurationOpen $theConfFile]

    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {

        # get the simulator:
        set theSimulator [openCarac_caracGetSimulator $theCarac]

        # reset to its default value:
        if { $theSimulator == $theDefaultSimulator } {
            openCarac_message "Simulator already is: $theDefaultSimulator"
        } else {
            openCarac_warning "Simulator was set to: $theSimulator."
            openCarac_message "Switch it to $theDefaultSimulator."
            openCarac_caracSetSimulator $theCarac $theDefaultSimulator
        }

    }

}
```

4.4.2.21 openCarac_caracMakeReadyForRunnings *theCarac*

Creates every openCarac *runnings*, children of openCarac *simulation*, for the whole openCarac *carac*.

For each possible corner/param/netlist triplet, creates an openCarac *running* child to each openCarac *simulation*. If open↩ Carac *application* "check mode" boolean is activated, number of openCarac *runnings* is reduced and only one openCarac *running* is kept per openCarac *simulation* name. Note that, in such a case, various openCarac *simulations* might have no openCarac *running* created. A list of every created openCarac *runnings* is returned. The lists of corner, param and netlist can be accessed through openCarac_caracGetCornerList, openCarac_caracGetParamList and openCarac_carac↩ GetNetlistList. The openCarac *simulations* list can be accessed through openCarac_caracGetSimulationsList. Application "check mode" value can be accessed through openCarac_applicationGetCheckMode. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_CARAC_MAKE_READY_FOR_RUNNINGS is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_CARAC_MAKE_READY_FOR_↩ RUNNINGS is executed.

Parameters

theCarac	: openCarac *carac*.

Returns

List ; openCarac *runnings*.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the model file must exist:
openCarac_caracSetModel  $theCarac "../../foundryModels/models.lib"
# the corners to load in the model file:
openCarac_caracAddCorner $theCarac "TYPICAL"
openCarac_caracAddCorner $theCarac "BEST"
openCarac_caracAddCorner $theCarac "WORST"

set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# there are three runnings:
set theNumberOfRunnings [llength $theRunningList]
puts "The number of runnings is: $theNumberOfRunnings"
```

4.4.2.22 openCarac_caracRemoveCorner *theCarac element*

Sets the value of "corner list" attribute of the openCarac *carac*.

Removes an element from the list of corners that are used to create openCarac *runnings* in the openCarac *carac* hierarchy. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a model file is loaded. To load the "model" file (which can be accessed through openCarac_caracGetModel), open↩ Carac adds it in the main file on the line following the "model marker" (accessible through openCarac_applicationGet↩ ModelMarker) after a "lib directive". The corner list is a list of strings to substitute after the model in the main file. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspice↩ GetLibDirective for ngspice). The element must not be a list itself or an empty string. An element cannot be removed if it is not already present in the corner list (case is not sensitive). The list can be accessed through openCarac_caracGet↩ CornerList.

Parameters

theCarac	: openCarac *carac*.
element	: String ; corner, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_caracAddCorner    $theCarac "TYPICAL"

# the corner list is not empty:
set numberOfCorners [llength [openCarac_caracGetCornerList $theCarac]]
puts "The number of corners is $numberOfCorners."

openCarac_caracRemoveCorner $theCarac "TYPICAL"

# the corner list is empty:
set numberOfCorners [llength [openCarac_caracGetCornerList $theCarac]]
puts "The number of corners is $numberOfCorners."
```

4.4.2.23 openCarac_caracRemoveExtractopFilter *theCarac element*

Sets the value of "extractop filter list" attribute of the openCarac *carac*.

Removes an element from the extractop filters list of the openCarac *carac*. If the "extractop filter list" is not empty, when parsing simulator files through openCarac_runningParseSimulatorFiles, instance parameters are extracted only if they match an element the "extractop filter list". Matching follows the rules of TCL "string match" command. It must be a string that is not empty and not a list itself. The element is not removed if not already available in the list. To define it, case sensitivity depends on the "case sensitivity" attribute of the selected simulator. The value of the selected simulator can be accessed through openCarac_caracGetSimulator. For more information about simulator case sensitivity, see access functions for attribute "case sensitivity" of the selected simulator (such as openCarac_ngspiceGetCaseSensitivity for ngspice). The list can be accessed through openCarac_caracGetExtractopFilterList.

Parameters

theCarac	: openCarac *carac*.
element	: String ; extractop filter, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_caracAddExtractopFilter    $theCarac "XDUT.M1"

# the extractop filter list is not empty:
set numberOfExtractopFilters [llength [openCarac_caracGetExtractopFilterList $theCarac]]
puts "The number of extractop filters is $numberOfExtractopFilters."

openCarac_caracRemoveExtractopFilter $theCarac "XDUT.M1"

# the extractop filter list is empty:
set numberOfExtractopFilters [llength [openCarac_caracGetExtractopFilterList $theCarac]]
puts "The number of extractop filters is $numberOfExtractopFilters."
```

4.4.2.24 openCarac_caracRemoveNetlist *theCarac element*

Sets the value of "netlist list" attribute of the openCarac *carac*.

Removes an element from the list of netlists that are used to create openCarac *runnings* in the openCarac *carac* hierarchy. The element must not be an empty string and must not contain hierarchy or extension. An element cannot be removed if it is not already present in the netlist list (case is sensitive). The list can be accessed through openCarac_caracGetNetlistList. Changing the netlist list affects the behaviour of openCarac_caracMakeReadyForRunnings.

Parameters

theCarac	: openCarac *carac*.
element	: String ; netlist, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_applicationSetNetlistFileExtension ".nsx"
# file myCircuit.nsx must exist:
openCarac_caracAddNetlist    $theCarac "myCircuit"

# the netlist list is not empty:
set numberOfNetlists [llength [openCarac_caracGetNetlistList $theCarac]]
puts "The number of netlists is $numberOfNetlists."

openCarac_caracRemoveNetlist $theCarac "myCircuit"

# the netlist list is empty:
set numberOfNetlists [llength [openCarac_caracGetNetlistList $theCarac]]
puts "The number of netlists is $numberOfNetlists."
```

4.4.2.25 openCarac_caracRemoveParam *theCarac element*

Sets the value of "param list" attribute of the openCarac *carac*.

Removes an element from the list of parameters that are used to create openCarac *runnings* in the openCarac *carac* hierarchy. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a libparam file is loaded. To load the "libparam" file (which can be accessed through openCarac_caracGet↩Libparam), openCarac adds it in the main file on the line following the "param marker" (accessible through openCarac↩_applicationGetParamMarker) after a "lib directive". The param list is a list of strings to substitute after the libparam in the main file. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). The element must not be a list itself or an empty string. An element cannot be removed if it is not already present in the param list (case is not sensitive). The list can be accessed through openCarac_caracGetParamList.

Parameters

theCarac	: openCarac *carac*.
element	: String ; param, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

openCarac_caracAddParam    $theCarac "TYPICAL"

# the param list is not empty:
set numberOfParams [llength [openCarac_caracGetParamList $theCarac]]
puts "The number of params is $numberOfParams."

openCarac_caracRemoveParam $theCarac "TYPICAL"

# the param list is empty:
set numberOfParams [llength [openCarac_caracGetParamList $theCarac]]
puts "The number of params is $numberOfParams."
```

4.4.2.26 openCarac_caracSetCheckmeas *theCarac name minValue maxValue*

Sets the value of "checkmeas list" attribute of the openCarac *carac*.

If a checkmeas with the same name is already present in the checkmeas list, it overwrites its minimum and maximum values ; otherwise it adds a new checkmeas to the checkmeas list. Every checkmeas element is a list itself, composed of three elements: the first one is its name, the second one is its minimum value and the third one is its maximum value. Its name must be a string that is not empty and not a list itself. Its minimum value and maximum values must be both doubles, both different from "NaN" ; minimum value must be inferior or equal to maximum value. Checkmeas list can be accessed through openCarac_caracGetCheckmeasList. Changing the checkmeas list affects the behaviour of openCarac_caracExtractResults and openCarac_runningExtractResults. It also has an impact on openCarac_configurationCreateArchives and openCarac_applicationCreateFullSummaryFile. The list can be accessed through openCarac_caracGetCheckmeasList.

Parameters

theCarac	: openCarac *carac*.
name	: String ; checkmeas name, non-empty, not a list itself.
minValue	: Double ; different from "NaN" and inferior or equal to maxValue.
maxValue	: Double ; different from "NaN" and superior or equal to minValue.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# checkmeas definition:
openCarac_caracSetCheckmeas $theCarac "PHASE_MARGIN" 60 180
openCarac_caracSetCheckmeas $theCarac "DC_GAIN"      42 Inf

# access their values:
foreach theCheckmeas [openCarac_caracGetCheckmeasList $theCarac] {
    set theName     [lindex $theCheckmeas 0]
    set theMinValue [lindex $theCheckmeas 1]
    set theMaxValue [lindex $theCheckmeas 2]

    puts "specification: $theMinValue < $theName < $theMaxValue"
}
```

4.4.2.27 openCarac_caracSetCheckop *theCarac name minValue maxValue*

Sets the value of "checkop list" attribute of the openCarac *carac*.

If a checkop with the same name is already present in the checkop list, it overwrites its minimum and maximum values ; otherwise it adds a new checkop to the checkop list. Every checkop element is a list itself, composed of three elements: the first one is its name, the second one is its minimum value and the third one is its maximum value. Its name must be a string that is not empty and not a list itself. Its minimum value and maximum values must be both doubles, both different from "NaN" ; minimum value must be inferior or equal to maximum value. Checkop list can be accessed through openCarac_caracGetCheckopList. Changing the checkop list affects the behaviour of openCarac_runningParseSimulatorFiles, openCarac_caracExtractResults and openCarac_runningExtractResults. It also has an impact on openCarac_configurationCreateArchives and openCarac_applicationCreateFullSummaryFile. The list can be accessed through openCarac_caracGetCheckopList.

Parameters

theCarac	: openCarac *carac*.
name	: String ; checkop name, non-empty, not a list itself.
minValue	: Double ; different from "NaN" and inferior or equal to maxValue.
maxValue	: Double ; different from "NaN" and superior or equal to minValue.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# checkop definition:
openCarac_caracSetCheckop $theCarac "OUTPUT" 3   3.3
openCarac_caracSetCheckop $theCarac "INPUT"  1.5 1.8

# access their values:
foreach theCheckop [openCarac_caracGetCheckopList $theCarac] {
    set theName     [lindex $theCheckop 0]
    set theMinValue [lindex $theCheckop 1]
    set theMaxValue [lindex $theCheckop 2]

    puts "specification: $theMinValue < V($theName) < $theMaxValue"
}
```

4.4.2.28 openCarac_caracSetCustomArgs *theCarac value*

Sets the value of "custom args" attribute of the openCarac *carac*.

Custom args remain unused by openCarac in most situations but it can be accessed by the user in a hook or a custom procedure (such as openCaracHook_ON_POST_RUNNING_EXECUTE_SIMULATOR or openCarac_customRunSimulator) to change a behaviour function of an openCarac *carac*. Its value can be accessed through openCarac_caracGetCustomArgs.

Parameters

theCarac	: openCarac *carac*.
value	: String ; custom args.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# set a list as custom args:
openCarac_caracSetCustomArgs $theCarac [list a 1 b 2 c 3]
```

4.4.2.29 openCarac_caracSetLibparam *theCarac value*

Sets the value of "libparam" attribute of the openCarac *carac*.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a libparam file is loaded. To load the "libparam" file, openCarac adds it in the main file on the line following the "param marker" (accessible through openCarac_applicationGetParamMarker) after a "lib directive". It also adds a "param" at the end of the "libparam" file loading. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). For more informations about "param" attribute, see access function openCarac_caracGetParamList. The libparam is either an empty string (no libparam to substitute) or a path to access the existing libparam file. This path can be absolute or relative to the parent openCarac *configuration* file. Configuration file path can be accessed through openCarac_configurationGetFilePath. Its value can be accessed through openCarac_caracGetLibparam.

Parameters

theCarac	: openCarac *carac*.
value	: String ; libparam file path.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# the libparam file must exist:
openCarac_caracSetLibparam $theCarac "../../parameters/conditions.lib"

set theLibparam [openCarac_caracGetLibparam $theCarac]
puts "The libparam is now set to: $theLibparam"
```

4.4.2.30 openCarac_caracSetModel *theCarac value*

Sets the value of "model" attribute of the openCarac *carac*.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and a model file is loaded. To load the "model" file, openCarac adds it in the main file on the line following the "model marker" (accessible through openCarac_applicationGetModelMarker) after a "lib directive". It also adds a "corner" at the end of the "model" file loading. For more informations about simulator "lib directive", see access function of the selected simulator (such as openCarac_ngspiceGetLibDirective for ngspice). For more informations about "corner" attribute, see access function openCarac_caracGetCornerList. The model is either an empty string (no model to substitute) or a path to access the existing model file. This path can be absolute or relative to the parent openCarac *configuration* file. Configuration file path can be accessed through openCarac_configurationGetFilePath. Its value can be accessed through openCarac_caracGetModel.

Parameters

theCarac	: openCarac *carac*.
value	: String ; model file path.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# the model file must exist:
openCarac_caracSetModel $theCarac "../../foundryModels/models.lib"

set theModel [openCarac_caracGetModel $theCarac]
puts "The model is now set to: $theModel"
```

4.4.2.31 openCarac_caracSetName *theCarac value*

Sets the value of "name" attribute of the openCarac *carac*.

Its value must be a string that is not empty and not a list itself. It must be formated to be a file name. It is not possible to have various openCarac *caracs* with the same name in an openCarac *configuration*. Its value can be accessed through openCarac_caracGetName.

Parameters

theCarac	: openCarac *carac*.
value	: String ; openCarac *carac* name.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# change the carac name:
```

```
openCarac_caracSetName $theCarac "aBrandNewName"

# access it:
puts "The carac name is: [openCarac_caracGetName $theCarac]"
```

4.4.2.32 openCarac_caracSetSimulator *theCarac value*

Sets the value of "simulator" attribute of the openCarac *carac*.

Changes the selected simulator for every openCarac *running* in the openCarac *carac* hierarchy. Selecting a simulator affects the behaviour of openCarac_runningExecuteSimulator and openCarac_runningParseSimulatorFiles. Its value can be accessed through openCarac_caracGetSimulator.

Parameters

theCarac	: openCarac *carac*.
value	: String ; name of the simulator.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# get the default simulator:
set theDefaultSimulator [openCarac_applicationGetDefaultSimulator]

# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {

    set theConf [openCarac_configurationOpen $theConfFile]

    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {

        # get the simulator:
        set theSimulator [openCarac_caracGetSimulator $theCarac]

        # reset to its default value:
        if { $theSimulator == $theDefaultSimulator } {
            openCarac_message "Simulator already is: $theDefaultSimulator"
        } else {
            openCarac_warning "Simulator was set to: $theSimulator."
            openCarac_message "Switch it to $theDefaultSimulator."
            openCarac_caracSetSimulator $theCarac $theDefaultSimulator
        }

    }

}
```

4.4.2.33 openCarac_caracUnsetCheckmeas *theCarac name*

Sets the value of "checkmeas list" attribute of the openCarac *carac*.

Removes a checkmeas from the checkmeas list of the openCarac *carac* function of its name. Its name must be a string that is not empty and not a list itself. Checkmeas list can be accessed through openCarac_caracGetCheckmeasList. Changing the checkmeas list affects the behaviour of openCarac_caracExtractResults and openCarac_runningExtractResults. It also has an impact on openCarac_configurationCreateArchives and openCarac_applicationCreateFullSummaryFile. The list can be accessed through openCarac_caracGetCheckmeasList.

Parameters

theCarac	: openCarac *carac*.
name	: String ; checkmeas name, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# checkmeas definition:
openCarac_caracSetCheckmeas $theCarac "PHASE_MARGIN" 60   180
openCarac_caracSetCheckmeas $theCarac "DC_GAIN"      42   Inf
openCarac_caracSetCheckmeas $theCarac "WRONG_NAME"   -Inf 0

# unset one:
openCarac_caracUnsetCheckmeas $theCarac "WRONG_NAME"

# there are two checkmeas:
set numberOfCheckmeas [llength [openCarac_caracGetCheckmeasList $theCarac]]
puts "The number of checkmeas is: $numberOfCheckmeas."
```

4.4.2.34 openCarac_caracUnsetCheckop *theCarac name*

Sets the value of "checkop list" attribute of the openCarac *carac*.

Removes a checkop from the checkop list of the openCarac *carac* function of its name. Its name must be a string that is not empty and not a list itself. Checkop list can be accessed through openCarac_caracGetCheckopList. Changing the checkop list affects the behaviour of openCarac_runningParseSimulatorFiles, openCarac_caracExtractResults and openCarac_runningExtractResults. It also has an impact on openCarac_configurationCreateArchives and openCarac_applicationCreateFullSummaryFile. The list can be accessed through openCarac_caracGetCheckopList.

Parameters

theCarac	: openCarac *carac*.
name	: String ; checkop name, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]

# checkop definition:
openCarac_caracSetCheckop $theCarac "OUTPUT"       3   3.3
openCarac_caracSetCheckop $theCarac "INPUT"        1.5 1.8
openCarac_caracSetCheckop $theCarac "WRONG_NAME"   0   1.8

# unset one:
openCarac_caracUnsetCheckop $theCarac "WRONG_NAME"

# there are two checkop:
set numberOfCheckop [llength [openCarac_caracGetCheckopList $theCarac]]
puts "The number of checkop is: $numberOfCheckop."
```

4.5 Simulation class

Definition of functions to manipulate openCarac *simulations*.

Functions

- openCarac_simulationCreate theParentcarac theName

  Creates an empty openCarac simulation.
- openCarac_simulationDelete theSimulation

  Deletes an openCarac simulation.
- openCarac_simulationGetParentCarac theSimulation

  Returns the value of "parent openCarac carac" attribute of the openCarac simulation.
- openCarac_simulationGetName theSimulation

  Returns the value of "name" attribute of the openCarac simulation.
- openCarac_simulationGetFilePath theSimulation

  Returns the value of "file path" attribute of the openCarac simulation.
- openCarac_simulationGetParameterList theSimulation

  Returns the value of "parameter list" attribute of the openCarac simulation.
- openCarac_simulationSetParameter theSimulation name value

  Sets the value of "parameter list" attribute of the openCarac simulation.
- openCarac_simulationUnsetParameter theSimulation name

  Sets the value of "parameter list" attribute of the openCarac simulation.
- openCarac_simulationGetExtractopList theSimulation

  Returns the value of "extractop list" attribute of the openCarac simulation.
- openCarac_simulationAddExtractop theSimulation name

  Sets the value of "extractop list" attribute of the openCarac simulation.
- openCarac_simulationRemoveExtractop theSimulation name

  Sets the value of "extractop list" attribute of the openCarac simulation.
- openCarac_simulationGetRunningsList theSimulation

  Returns the value of "runnings list" attribute of the openCarac simulation.

4.5.1 Detailed Description

Definition of functions to manipulate openCarac *simulations*.

Here are defined every API functions that are used to access openCarac *simulations*. Various openCarac *simulations* can be defined per openCarac *carac*. An openCarac *simulation* contains a list of Spice parameters to sweep and a list of Spice parameters to extract at operating point. Each openCarac *simulation* may have various openCarac *running* children.

4.5.2 Function Documentation

4.5.2.1 openCarac_simulationAddExtractop *theSimulation name*

Sets the value of "extractop list" attribute of the openCarac *simulation*.

Its name must be a string that is not empty and not a list itself. If an extractop with the same name is already present in the extractops list, it is not added again. To define it, case sensitivity depends on the "case sensitivity" attribute of the selected simulator. The value of the selected simulator can be accessed through openCarac_caracGetSimulator. For more information about simulator case sensitivity, see access functions for attribute "case sensitivity" of the selected simulator (such as openCarac_ngspiceGetCaseSensitivity for ngspice). Changing the extractop list affects the behaviour of openCarac_runningParseSimulatorFiles. For each element of this list, when calling the appropriate simulator files parser to extract measure values, openCarac also looks for instance parameters at operating point. If a parameter matches an extractop name, a measure is set to the openCarac *running* with its name as parameterName(instanceName) and its value as the parameter value. This has the same effect as calling openCarac_runningSetMeasure. Instance names can be filtered when matching one element of the "extractop filter list" attribute of the parent openCarac *carac*, its value can be accessed through openCarac_caracGetExtractopFilterList. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. The list can be accessed through openCarac_simulationGetExtractopList.

Parameters

theSimulation	: openCarac *simulation*.
name	: String ; extractop, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

openCarac_simulationAddExtractop $theSimu "IN"
openCarac_simulationAddExtractop $theSimu "OUT"

# there are two net voltages to extract:
set theNumberOfExtractop [llength [openCarac_simulationGetExtractopList $theSimu]]
puts "The number of extractop is $theNumberOfExtractop"
```

4.5.2.2 openCarac_simulationCreate *theParentcarac theName*

Creates an empty openCarac *simulation*.

Adds this new openCarac *simulation* to the hierarchy of the parent openCarac *carac*. The openCarac *simulation* name must not be an empty string and must not contain hierarchy or extension. In order to create an openCarac *simulation*, make sure that the file exists with the appropriate extension (see openCarac_applicationGetSimulationFileExtension for more informations) in the same directory as the main file ; i.e. next to the parent openCarac *configuration* file. Parent openCarac *configuration* can be accessed through openCarac_caracGetParentConfiguration and openCarac *configuration* file path through openCarac_configurationGetFilePath. List of available openCarac *simulations* in the parent openCarac *carac* can be accessed through openCarac_caracGetSimulationsList.

Parameters

theParentcarac	: openCarac *carac*.
theName	: String ; non-empty, a file name with no hierarchy and no extension.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the simulation is available:
puts "This is the simulation name: [openCarac_simulationGetName $theSimu]"
```

4.5.2.3 openCarac_simulationDelete *theSimulation*

Deletes an openCarac *simulation*.

Frees the memory from the informations about the openCarac *simulation*. After calling this function, the openCarac *simulation* is not available in the list returned by openCarac_caracGetSimulationsList and every openCarac *running* children is not available.

Parameters

theSimulation	: openCarac *simulation*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the simulation is available:
set theSimulationName [openCarac_simulationGetName $theSimu]
puts "This is the simulation name: $theSimulationName"

# delete the simulation:
openCarac_simulationDelete $theSimu

# the simulation is no longer available:
set theCode [openCarac_simulationGetName $theSimu]
puts "The returned code is: $theCode"
```

4.5.2.4 openCarac_simulationGetExtractopList *theSimulation*

Returns the value of "extractop list" attribute of the openCarac *simulation*.

Every extractop is a string that is not empty and not a list itself. Changing the extractop filter list affects the behaviour of openCarac_runningParseSimulatorFiles. For each element of this list, when calling the appropriate simulator files parser to extract measure values, openCarac also looks for instance parameters at operating point. If a parameter matches an extractop name, a measure is set to the openCarac *running* with its name as parameterName(instanceName) and its value as the parameter value. This has the same effect as calling openCarac_runningSetMeasure. Instance names can be filtered when matching one element of the "extractop filter list" attribute of the parent openCarac *carac*, its value can be accessed through openCarac_caracGetExtractopFilterList. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Extractops can be set or unset through openCarac_simulationAddExtractop and openCarac_simulationRemoveExtractop.

Parameters

theSimulation	: openCarac *simulation*.

Returns

List ; Strings that are not empty and not lists themselves.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

openCarac_simulationAddExtractop $theSimu "IN"
openCarac_simulationAddExtractop $theSimu "OUT"

# there are two net voltages to extract:
set theNumberOfExtractop [llength [openCarac_simulationGetExtractopList $theSimu]]
puts "The number of extractop is $theNumberOfExtractop"
```

4.5.2.5 openCarac_simulationGetFilePath *theSimulation*

Returns the value of "file path" attribute of the openCarac *simulation*.

It is a normalized absolute path to access the file that can be included in the Spice main file. This is the file that matches the openCarac *simulation* name (used for its creation with openCarac_simulationCreate).

Parameters

theSimulation	: openCarac *simulation*.

Returns

String ; openCarac *simulation* file path ; integer -1 if an error occurred.

Example

```
# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {

    set theConf [openCarac_configurationOpen $theConfFile]

    # access the caracs:
    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {
        # access the simulations:
        foreach theSimulation [openCarac_caracGetSimulationsList $theCarac] {
            # print their file path:
            set theSimuFile [openCarac_simulationGetFilePath $theSimulation]
            puts "This is a simulation file: $theSimuFile"
        }
    }

}
```

4.5.2.6 openCarac_simulationGetName *theSimulation*

Returns the value of "name" attribute of the openCarac *simulation*.

This name is the one that has been used when creating the openCarac *simulation* through openCarac_simulationCreate. It is the tail (with no extension) of the file that can be included in the Spice main file. Its normalized absolute path can be accessed through openCarac_simulationGetFilePath.

Parameters

theSimulation	: openCarac *simulation*.

Returns

String ; openCarac *simulation* name ; integer -1 if an error occurred.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the simulation is available:
puts "This is the simulation name: [openCarac_simulationGetName $theSimu]"
```

4.5.2.7 openCarac_simulationGetParameterList *theSimulation*

Returns the value of "parameter list" attribute of the openCarac *simulation*.

Each element of this list is a list itself, composed of two elements: the first one is its name, the second one is its value. Its name is a string that is not empty and not a list itself. Its value is a string that is not empty. Changing the parameter list affects the behaviour of openCarac_runningCreateTemporaryFolder : it changes what to substitute in the files when creating temporary folders. It also has an impact on openCarac_caracExtractResults, openCarac_runningExtractResults, openCarac_configurationCreateArchives and openCarac_applicationCreateFullSummaryFile. Elements can be added or removed through openCarac_simulationSetParameter and openCarac_simulationUnsetParameter.

Parameters

theSimulation	: openCarac *simulation*.

Returns

List ; Two strings that are not empty, the first one is not a list itself.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# set the parameters:
openCarac_simulationSetParameter $theSimu "myParam"      16
openCarac_simulationSetParameter $theSimu "myOtherParam" 42

# unset one:
openCarac_simulationUnsetParameter $theSimu "myOtherParam"

# access the list:
foreach theParameter [openCarac_simulationGetParameterList $theSimu] {
    set theName  [lindex $theParameter 0]
    set theValue [lindex $theParameter 1]
    puts "Parameter $theName must be set to: $theValue"
}
```

4.5.2.8 openCarac_simulationGetParentCarac *theSimulation*

Returns the value of "parent openCarac *carac"* attribute of the openCarac *simulation*.

Parameters

theSimulation	: openCarac *simulation*.

Returns

openCarac *carac* ; integer -1 if an error occurred.

Example

```
# access data in the custom procedure:
proc openCarac_customRunSimulator { theMainFilePath } {
    # get the running:
    set theRunning    [openCarac_runningGetFromMainFilePath $theMainFilePath]
    # get the running parent hierarchy:
    set theSimulation [openCarac_runningGetParentSimulation $theRunning]
    set theCarac      [openCarac_simulationGetParentCarac   $theSimulation]

    # the main code here...
}
```

4.5.2.9 openCarac_simulationGetRunningsList *theSimulation*

Returns the value of "runnings list" attribute of the openCarac *simulation*.

These are the openCarac *running* children. An openCarac *running* can be loaded by making an openCarac *carac* ready for openCarac *runnings* (through openCarac_caracMakeReadyForRunnings) : this loads in the openCarac *simulation* an openCarac *running* for each possible triplet of corner/param/netlist.

Parameters

theSimulation	: openCarac *simulation*.

Returns

List ; openCarac *runnings* ; integer -1 if an error occurred.

Example

```
# open existing configurations:
foreach theConfFile [openCarac_applicationGetConfigurationFileList] {

    set theConf [openCarac_configurationOpen $theConfFile]

    # access the caracs:
    foreach theCarac [openCarac_configurationGetCaracsList $theConf] {

        openCarac_caracMakeReadyForRunnings $theCarac

        # access the simulations:
        foreach theSimulation [openCarac_caracGetSimulationsList $theCarac] {
            # access the runnings:
            foreach theRunning [openCarac_simulationGetRunningsList $theSimulation] {
                set theMainFile [openCarac_runningGetMainFilePath $theRunning]
                puts "The main file is: $theMainFile"
            }
        }
    }

}
```

4.5.2.10 openCarac_simulationRemoveExtractop *theSimulation name*

Sets the value of "extractop list" attribute of the openCarac *simulation*.

Removes an element in the extractop list of the openCarac *simulation* function of its name. Its name must be a string that is not empty and not a list itself. An extractop cannot be removed if it is not already present in the extractop list. To define it, case sensitivity depends on the "case sensitivity" attribute of the selected simulator. The value of the selected simulator can be accessed through openCarac_caracGetSimulator. For more information about simulator case sensitivity, see access functions for attribute "case sensitivity" of the selected simulator (such as openCarac_ngspiceGetCaseSensitivity for ngspice). Changing the extractop list affects the behaviour of openCarac_runningParseSimulatorFiles. For each element of this list, when calling the appropriate simulator files parser to extract measure values, openCarac also looks for instance parameters at operating point. If a parameter matches an extractop name, a measure is set to the openCarac *running* with its name as parameterName(instanceName) and its value as the parameter value. This has the same effect as calling openCarac_runningSetMeasure. Instance names can be filtered when matching one element of the "extractop filter list" attribute of the parent openCarac *carac*, its value can be accessed through openCarac_caracGetExtractopFilterList. The list can be accessed through openCarac_simulationGetExtractopList.

Parameters

theSimulation	: openCarac *simulation*.
name	: String ; extractop, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

openCarac_simulationAddExtractop $theSimu "IN"
openCarac_simulationAddExtractop $theSimu "OUT"

# remove one net voltage to extract:
openCarac_simulationRemoveExtractop $theSimu "IN"

# there is one net voltage to extract:
set theNumberOfExtractop [llength [openCarac_simulationGetExtractopList $theSimu]]
puts "The number of extractop is $theNumberOfExtractop"
```

4.5.2.11 openCarac_simulationSetParameter *theSimulation name value*

Sets the value of "parameter list" attribute of the openCarac *simulation*.

Adds the setting of a parameter, function of its name, in the parameters list of the openCarac *simulation*. Its name must be a string that is not empty and not a list itself. Its value must a string that is not empty. If a parameter with the same name is already present in the parameters list, it overwrites its value ; otherwise it adds a new parameter to the parameters list. To define it, case sensitivity depends on the "case sensitivity" attribute of the selected simulator. The value of the selected simulator can be accessed through openCarac_caracGetSimulator. For more information about simulator case sensitivity, see access functions for attribute "case sensitivity" of the selected simulator (such as open↩ Carac_ngspiceGetCaseSensitivity for ngspice). Changing the parameter list affects the behaviour of openCarac_running↩ CreateTemporaryFolder : it changes what to substitute in the files when creating temporary folders. It also has an impact on openCarac_caracExtractResults, openCarac_runningExtractResults, openCarac_configurationCreateArchives and open↩ Carac_applicationCreateFullSummaryFile. The list can be accessed through openCarac_simulationGetParameterList.

Parameters

theSimulation	: openCarac *simulation*.
name	: String ; parameter name, non-empty, not a list itself.
value	: String ; parameter value, non-empty.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# set the parameters:
openCarac_simulationSetParameter $theSimu "myParam"      16
openCarac_simulationSetParameter $theSimu "myOtherParam" 42

# unset one:
openCarac_simulationUnsetParameter $theSimu "myOtherParam"

# access the list:
foreach theParameter [openCarac_simulationGetParameterList $theSimu] {
    set theName  [lindex $theParameter 0]
    set theValue [lindex $theParameter 1]
    puts "Parameter $theName must be set to: $theValue"
}
```

4.5.2.12 openCarac_simulationUnsetParameter *theSimulation name*

Sets the value of "parameter list" attribute of the openCarac *simulation*.

Removes the setting of a parameter, function of its name, in the parameters list of the openCarac *simulation*. Its name must be a string that is not empty and not a list itself. A parameter cannot be unset if it is not already present in the parameter list. To define it, case sensitivity depends on the "case sensitivity" attribute of the selected simulator. The value of the selected simulator can be accessed through openCarac_caracGetSimulator. For more information about simulator case sensitivity, see access functions for attribute "case sensitivity" of the selected simulator (such as openCarac_ngspiceGetCaseSensitivity for ngspice). Changing the parameter list affects the behaviour of openCarac_runningCreateTemporaryFolder : it changes what to substitute in the files when creating temporary folders. It also has an impact on openCarac_caracExtractResults, openCarac_runningExtractResults, openCarac_configurationCreateArchives and openCarac_applicationCreateFullSummaryFile. The list can be accessed through openCarac_simulationGetParameterList.

Parameters

theSimulation	: openCarac *simulation*.
name	: String ; parameter name, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# set the parameters:
openCarac_simulationSetParameter $theSimu "myParam"      16
openCarac_simulationSetParameter $theSimu "myOtherParam" 42

# unset one:
openCarac_simulationUnsetParameter $theSimu "myOtherParam"

# access the list:
foreach theParameter [openCarac_simulationGetParameterList $theSimu] {
    set theName  [lindex $theParameter 0]
    set theValue [lindex $theParameter 1]
    puts "Parameter $theName must be set to: $theValue"
}
```

4.6 Running class

Definition of functions to manipulate openCarac *runnings*.

Functions

- openCarac_runningGetParentSimulation theRunning

    Returns the value of "parent openCarac simulation" attribute of the openCarac running.
- openCarac_runningGetCorner theRunning

    Returns the value of "corner" attribute of the openCarac running.
- openCarac_runningGetParam theRunning

    Returns the value of "param" attribute of the openCarac running.
- openCarac_runningGetNetlist theRunning

    Returns the value of "netlist" attribute of the openCarac running.
- openCarac_runningGetFromMainFilePath theMainFilePath

    Function to find a specific openCarac running knowing the main file path in the temporary folder.
- openCarac_runningGetMainFilePath theRunning

    Returns the value of "main file path" attribute of the openCarac running.
- openCarac_runningGetSimulationIncludedFilePath theRunning

    Returns the value of "simulation included file path" attribute of the openCarac running.
- openCarac_runningGetSimulatorFilesSavingFolderPath theRunning

    Returns the value of "simulator files saving folder path" attribute of the openCarac running.
- openCarac_runningCreateTemporaryFolder theRunning

    Creates the temporary folder for a specific openCarac running whilst modifying the copied files.
- openCarac_runningExecuteSimulator theRunning

    Executes the appropriate command of the simulator.
- openCarac_runningParseSimulatorFiles theRunning

    Calls the openCarac simulator files parser for the selected simulator.
- openCarac_runningDeleteTemporaryFolder theRunning

    Deletes the openCarac running temporary folder.
- openCarac_runningSetMeasure theRunning theResultStep theResultName theResultValue

    Sets the value of "measure" attribute of the openCarac running.
- openCarac_runningUnsetMeasure theRunning theResultStep theResultName

    Sets the value of "measure" attribute of the openCarac running.
- openCarac_runningClearResults theRunning

    Clear the openCarac running of every measure, openCarac resultstructure and openCarac result.
- openCarac_runningSaveResults theRunning

    Write the values of the openCarac running measures into a file to be used for results extraction.
- openCarac_runningExtractResults theRunning

    Extract results of the openCarac running.
- openCarac_runningGetResultstructuresList theRunning

    Returns the value of "resultstructures list" attribute of the openCarac running.

4.6.1 Detailed Description

Definition of functions to manipulate openCarac *runnings*.

Here are defined every API functions that are used to access openCarac *runnings*. Various openCarac *runnings* can be defined per openCarac *simulation*. An openCarac *running* contains informations about one running of the Spice simulator in a temporary folder created for this purpose. Each openCarac *running* may have various openCarac *resultstructure* children.

4.6.2 Function Documentation

4.6.2.1 openCarac_runningClearResults *theRunning*

Clear the openCarac *running* of every measure, openCarac *resultstructure* and openCarac *result*.

Unsets the name, step and value for every measure to an openCarac *running*. Also, after executing this command, openCarac *resultstructure* and openCarac *result* children are no longer available. Measures can be set to the openCarac *running* through openCarac_runningSetMeasure. openCarac *resultstructure* and openCarac *result* are created when calling openCarac_runningExtractResults.

Parameters

theRunning	: openCarac *running*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# extract the results:
foreach theRunning $theRunningList {

    openCarac_runningSetMeasure $theRunning "step 1" "MY_MEASURE"    42
    openCarac_runningSetMeasure $theRunning "step 1" "OTHER_MEASURE" 16

    # extract the results:
    openCarac_runningSaveResults    $theRunning
    openCarac_runningExtractResults $theRunning

    # two structures are available:
    set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
    puts "The number of structures is: [llength $theStructureList]"

    openCarac_runningClearResults $theRunning

    # no structure is available:
    set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
    puts "The number of structures is: [llength $theStructureList]"
}
```

4.6.2.2 openCarac_runningCreateTemporaryFolder *theRunning*

Creates the temporary folder for a specific openCarac *running* whilst modifying the copied files.

When loading the parent openCarac *configuration*, openCarac parses every file having an extension matching one of the openCarac files extension filter (this filter can be accessed through openCarac_applicationGetFilesExtensionFilter) that is found in its directory ; such files are buffered and their text are used to create the temporary folder. To execute the simulator, a temporary folder is created where the files are re-written and substitutions occur. For each substitution, case sensitivity is defined by the selected simulator "case sensitivity" attribute. Simulator "lib directive", model file path and corner value are substituted after the model marker. Simulator "lib directive", libparam file path and param value are substituted after the param marker. Simulator "inc directive", parent openCarac *simulation* name and simulation file extension are substituted after the simu marker. Simulator "inc directive", netlist value and netlist file extension are substituted on the line starting with the simulator "inc directive" and containing the netlist file extension. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. For each parameter set in the parent openCarac *simulation*, simulator "param directive", parameter name and parameter value are substituted on the lines starting with the simulator "param directive" and containing the parameter name. If openCarac *application* "comment of possible inclusions" boolean is activated, simulator comment syntax is added at the beginning of any line starting with the simulator "inc directive" and containing a possible openCarac *simulation* name. Also, simulator "comment syntax" is added at the beginning of any line starting with the simulator "lib directive" and containing a possible "model" file tail or a possible "libparam" file tail. Possible openCarac *simulation* names and "model" or "libparam" file tails are defined from every openCarac *carac*, every openCarac *simulation* in the hierarchy of the parent openCarac *configuration*. In case openCarac *application* "check mode" boolean is activated, simulator "comment syntax" is added at the beginning of any line starting with a pattern of the simulator "to remove in check mode" list. For more informations about simulator "case sensitivity", "lib directive", "inc directive", "string delimiter", "command syntax" or "to remove in check mode", see access functions of the selected simulator (such as openCarac_ngspiceGetCaseSensitivity, openCarac_ngspiceGetLibDirective, openCarac_ngspiceGetIncDirective, openCarac_ngspiceGetStringDelimiter, openCarac_ngspiceGetCommentSyntax or openCarac_ngspiceGetToRemoveInCheckMode for ngspice). For more informations about model file path or libparam file path see access functions for openCarac *carac* attributes "model" and "libparam": openCarac_caracGetModel and openCarac_caracGetLibparam. For more informations about corner value, param value and netlist value, see access functions for openCarac *running* attributes "corner", "param" and "netlist": openCarac_runningGetCorner, openCarac_runningGetParam and openCarac_runningGetNetlist. Markers can be accessed through openCarac_applicationGetModelMarker, openCarac_applicationGetParamMarker and openCarac_applicationGetSimuMarker. Simulation name can be accessed through openCarac_simulationGetName. Simulation parameters list can be accessed through openCarac_simulationGetParameterList. Simulation and netlist files extensions can be accessed through openCarac_applicationGetSimulationFileExtension and openCarac_applicationGetNetlistFileExtension. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_RUNNING_CREATE_TEMPORARY_FOLDER is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_RUNNING_CREATE_TEMPORARY_FOLDER is executed.

Parameters

theRunning	: openCarac *running*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the model file must exist:
openCarac_caracSetModel  $theCarac "../../foundryModels/models.lib"
openCarac_caracAddCorner $theCarac "BEST"
openCarac_caracAddCorner $theCarac "WORST"

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# create their temporary folders:
foreach theRunning $theRunningList {
    openCarac_runningCreateTemporaryFolder $theRunning
}
```

4.6.2.3 openCarac_runningDeleteTemporaryFolder *theRunning*

Deletes the openCarac *running* temporary folder.

Deletion of openCarac *running* temporary folder occurs only if openCarac *application* "debug mode" is deactivated ; otherwise, a warning is printed and the temporary folder is not deleted. The value of openCarac *application* "debug mode" boolean can be accessed through openCarac_applicationGetDebugMode. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_RUNNING_DELETE_TEMPORARY_FOLDER is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_RUNNING_DELETE_TEMPORARY_FOLDER is executed.

Parameters

theRunning	: openCarac *running*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the model file must exist:
openCarac_caracSetModel  $theCarac "../../foundryModels/models.lib"
openCarac_caracAddCorner $theCarac "BEST"
openCarac_caracAddCorner $theCarac "WORST"

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

foreach theRunning $theRunningList {
    openCarac_runningCreateTemporaryFolder $theRunning
    openCarac_runningExecuteSimulator      $theRunning
    openCarac_runningParseSimulatorFiles   $theRunning
    # after the execution the simulator and the parsing of the files:
    openCarac_runningDeleteTemporaryFolder $theRunning
}
```

4.6.2.4 openCarac_runningExecuteSimulator *theRunning*

Executes the appropriate command of the simulator.

Depending on the value of openCarac *application* "custom execution mode" boolean, either the default openCarac behaviour is called or the custom procedure is called. In both cases, a directory change to the temporary folder is performed if the openCarac simulator "directory change" attribute is activated ; in such a case, openCarac changes back to the original folder after the simulator execution. The custom execution is defined by openCarac_customRunSimulator. The default openCarac behaviour uses TCL "exec" command with, as arguments, the simulator command, the simulator option and the path to access the temporary main file. The simulator option is defined function of openCarac *application* "check mode" boolean. The execution output is redirected into a file having the same root name as the temporary main file path but with its extension set to the simulator "log extension". For more informations about openCarac *application* "custom execution mode" boolean, see access function openCarac_applicationGetCustomExecutionMode. For more informations about simulator "command", "run option", "check option", "directory change" or "log extension", see access functions of the selected simulator (such as openCarac_ngspiceGetCommand, openCarac_ngspiceGetRunOptions, openCarac_↩ ngspiceGetCheckOptions, openCarac_ngspiceGetDirectoryChange or openCarac_ngspiceGetLogExtension for ngspice). When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_RUNNING_EX↩ ECUTE_SIMULATOR is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_RUNNI↩ NG_EXECUTE_SIMULATOR is executed.

Parameters

theRunning	: openCarac *running*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the model file must exist:
openCarac_caracSetModel  $theCarac "../../foundryModels/models.lib"
openCarac_caracAddCorner $theCarac "BEST"
openCarac_caracAddCorner $theCarac "WORST"

# selection of the simulator to execute:
openCarac_caracSetSimulator $theCarac "ngspice"

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

foreach theRunning $theRunningList {
    openCarac_runningCreateTemporaryFolder $theRunning
    # execution of the simulator:
    openCarac_runningExecuteSimulator      $theRunning
    openCarac_runningParseSimulatorFiles   $theRunning
    openCarac_runningDeleteTemporaryFolder $theRunning
}
```

4.6.2.5 openCarac_runningExtractResults *theRunning*

Extract results of the openCarac *running*.

Depending on the values of openCarac *application* booleans "creation of html files", "creation of latex files" and "creation of octave files", creates output files for the openCarac *running*. Files are created in a "openCaracFiles" folder located next to the parent openCarac *configuration* file. Parent openCarac *configuration* can be accessed through openCarac↵_caracGetParentConfiguration and openCarac *configuration* file path through openCarac_configurationGetFilePath. Note that, first, measures must have been added to openCarac *runnings* (through openCarac_runningSetMeasure) and results must have been saved (through openCarac_runningSaveResults). Parsing simulator files in the temporary folder (with openCarac_runningParseSimulatorFiles) also does these two steps. When creating output files, openCarac makes special treatment for any result to be checked, i.e. any measure having its name matching a checkmeas name or matching a string starting with "V" and matching a checkop name between parentheses. Indeed, in output files, results values are checked function of checkmeas or checkop values. Matching follows the rules of TCL "string match" command. After calling this function, openCarac *resultstructure* and openCarac *result* children are available. Both checkmeas list and checkop list can be accessed through openCarac_caracGetCheckmeasList and openCarac_caracGetCheckopList. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_RUNNING_EXTRACT_RESULTS is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_RUNNING_EXTRACT_RESULTS is executed.

Parameters

theRunning	: openCarac *running*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# extract the results:
foreach theRunning $theRunningList {

    openCarac_runningSetMeasure $theRunning "step 1" "MY_MEASURE"    42
    openCarac_runningSetMeasure $theRunning "step 1" "OTHER_MEASURE" 16

    # extract the results:
    openCarac_runningSaveResults    $theRunning
    openCarac_runningExtractResults $theRunning

    # two structures are available:
    set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
    puts "The number of structures is: [llength $theStructureList]"
}
```

4.6.2.6 openCarac_runningGetCorner *theRunning*

Returns the value of "corner" attribute of the openCarac *running*.

When creating openCarac *runnings* through openCarac_caracMakeReadyForRunnings, each one of them sees itself affected a corner from the parent openCarac *carac* attribute "corner list" (accessible through openCarac_caracGetCornerList). This function returns the openCarac *running* affected "corner". The "corner" value is used when creating the openCarac *running* temporary folder through openCarac_runningCreateTemporaryFolder. If no corner is to be substituted, the corner value is an empty string.

Parameters

theRunning	: openCarac *running*.

Returns

String ; openCarac *running* "corner" value ; integer -1 if an error occurred.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the model file must exist:
openCarac_caracSetModel  $theCarac "../../foundryModels/models.lib"
# the corners to load in the model file:
openCarac_caracAddCorner $theCarac "BEST"
openCarac_caracAddCorner $theCarac "WORST"

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# the corner is either "BEST" or "WORST":
foreach theRunning $theRunningList {
    puts "The corner is: [openCarac_runningGetCorner $theRunning]"
}
```

4.6.2.7 openCarac_runningGetFromMainFilePath *theMainFilePath*

Function to find a specific openCarac *running* knowing the main file path in the temporary folder.

When executing the simulator (through openCarac_runningExecuteSimulator) while having "custom execution mode" activated (by openCarac_applicationActivateCustomExecutionMode), openCarac calls the custom procedure openCarac_customRunSimulator. The only available argument is the "main file path" in the temporary folder. In order to access attributes of the appropriate openCarac *running*, this function finds it from its "main file path" and returns it.

Parameters

theMainFilePath	: String ; openCarac *running* absolute "main file path".

Returns

openCarac *running* ; integer -1 if an error occurred.

Example

```
# access data in the custom procedure:
proc openCarac_customRunSimulator { theMainFilePath } {
    # get the running:
    set theRunning    [openCarac_runningGetFromMainFilePath $theMainFilePath]
    # get the running parent hierarchy:
    set theSimulation [openCarac_runningGetParentSimulation $theRunning]
    set theCarac      [openCarac_simulationGetParentCarac   $theSimulation]

    # the main code here...
}
```

4.6.2.8 openCarac_runningGetMainFilePath *theRunning*

Returns the value of "main file path" attribute of the openCarac *running*.

When calling the simulator for a specific openCarac *running* in a temporary folder created for this purpose, a main file path must be loaded by the simulator. This function returns a string: the absolute file path of the main file. This file is a modified copy of the identified main file when creating (through openCarac_configurationCreate) or opening (through openCarac_configurationOpen) an openCarac *configuration*. The path of the original main file can be accessed through openCarac_configurationGetMainFilePath. It is copied when creating temporary folder openCarac_runningCreateTemporaryFolder. The main file path is returned even if the temporary folder does not exist.

Parameters

theRunning	: openCarac *running*.

Returns

String ; openCarac *running* absolute "main file path" ; integer -1 if an error occurred.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the model file must exist:
openCarac_caracSetModel  $theCarac "../../foundryModels/models.lib"
openCarac_caracAddCorner $theCarac "BEST"
openCarac_caracAddCorner $theCarac "WORST"
```

```

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

foreach theRunning $theRunningList {
    # access informations:
    set theMainFile     [openCarac_runningGetMainFilePath $theRunning]
    set theIncludedFile [openCarac_runningGetSimulationIncludedFilePath $theRunning]

    puts "In the temporary folder:"
    puts "- the main file is: $theMainFile"
    puts "- the included file is: $theIncludedFile"

    openCarac_runningCreateTemporaryFolder $theRunning
}
```

4.6.2.9 openCarac_runningGetNetlist *theRunning*

Returns the value of "netlist" attribute of the openCarac *running*.

When creating openCarac *runnings* through openCarac_caracMakeReadyForRunnings, each one of them sees itself affected a netlist from the parent openCarac *carac* attribute "netlist list" (accessible through openCarac_caracGetNetlistList). This function returns the openCarac *running* affected "netlist". The "netlist" value is used when creating the openCarac *running* temporary folder through openCarac_runningCreateTemporaryFolder. If no netlist is to be substituted, the netlist value is an empty string.

Parameters

theRunning	: openCarac *running*.

Returns

String ; openCarac *running* "netlist" value ; integer -1 if an error occurred.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

openCarac_applicationSetNetlistFileExtension ".nsx"
# files myCircuit.nsx and myOtherCircuit.nsx must exist:
openCarac_caracAddNetlist $theCarac "myCircuit"
openCarac_caracAddNetlist $theCarac "myOtherCircuit"

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# the netlist is either "myCircuit" or "myOtherCircuit":
foreach theRunning $theRunningList {
    puts "The netlist is: [openCarac_runningGetNetlist $theRunning]"
}
```

4.6.2.10 openCarac_runningGetParam *theRunning*

Returns the value of "param" attribute of the openCarac *running*.

When creating openCarac *runnings* through openCarac_caracMakeReadyForRunnings, each one of them sees itself affected a param from the parent openCarac *carac* attribute "param list" (accessible through openCarac_caracGetParamList). This function returns the openCarac *running* affected "param". The "param" value is used when creating the openCarac *running* temporary folder through openCarac_runningCreateTemporaryFolder. If no param is to be substituted, the param value is an empty string.

Parameters

theRunning	: openCarac *running*.

Returns

String ; openCarac *running* "param" value ; integer -1 if an error occurred.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the libparam file must exist:
openCarac_caracSetLibparam $theCarac "../../parameters/conditions.lib"
openCarac_caracAddParam    $theCarac "BEST"
openCarac_caracAddParam    $theCarac "WORST"

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# the param is either "BEST" or "WORST":
foreach theRunning $theRunningList {
    puts "The param is: [openCarac_runningGetParam $theRunning]"
}
```

4.6.2.11 openCarac_runningGetParentSimulation *theRunning*

Returns the value of "parent openCarac *simulation"* attribute of the openCarac *running*.

Parameters

theRunning	: openCarac *running*.

Returns

openCarac *simulation* ; integer -1 if an error occurred.

Example

```
# access data in the custom procedure:
proc openCarac_customRunSimulator { theMainFilePath } {
    # get the running:
    set theRunning    [openCarac_runningGetFromMainFilePath $theMainFilePath]
    # get the running parent hierarchy:
    set theSimulation [openCarac_runningGetParentSimulation $theRunning]
    set theCarac      [openCarac_simulationGetParentCarac   $theSimulation]

    # the main code here...
}
```

4.6.2.12 openCarac_runningGetResultstructuresList *theRunning*

Returns the value of "resultstructures list" attribute of the openCarac *running*.

After having extracted the results, i.e. by calling openCarac_runningExtractResults, the previously set measures values are available and have been checked. These results are set in openCarac *resultstructures*. Measures can be set to the openCarac *running* through openCarac_runningSetMeasure or openCarac_runningParseSimulatorFiles. Measure values are checked function of the parent openCarac *carac* checkmeas list and checkop list, they can be accessed through open↩ Carac_caracGetCheckmeasList and openCarac_caracGetCheckopList.

Parameters

theRunning	: openCarac *running*.

Returns

List ; openCarac *resultstructures* ; integer -1 if an error occurred.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

foreach theRunning $theRunningList {

    openCarac_runningSetMeasure $theRunning "step 1" "MY_MEASURE"    42
    openCarac_runningSetMeasure $theRunning "step 1" "OTHER_MEASURE" 16

    # extract the results:
    openCarac_runningSaveResults    $theRunning
    openCarac_runningExtractResults $theRunning

    # the structures are available:
    set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
    foreach theStructure $theStructureList {
        set theName [openCarac_resultstructureGetName $theStructure]
        puts "A result structure has been extracted: $theName"
    }
}
```

4.6.2.13 openCarac_runningGetSimulationIncludedFilePath *theRunning*

Returns the value of "simulation included file path" attribute of the openCarac *running*.

When calling the simulator for a specific openCarac *running* in a temporary folder created for this purpose, the main file is copied and modified to include a modified openCarac *simulation* file. This function returns a string: the absolute file path of the openCarac *simulation* included file. This file is a modified copy of the file that matches the parent openCarac *simulation* name (used for its creation with openCarac_simulationCreate). The path of the original file can be accessed through openCarac_simulationGetFilePath. It is copied when creating temporary folder openCarac_runningCreateTemporary↩Folder. The openCarac *simulation* included file path is returned even if the temporary folder does not exist.

Parameters

theRunning	: openCarac *running*.

Returns

String ; openCarac *running* absolute "simulation included file path" ; integer -1 if an error occurred.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]
```

```

# the model file must exist:
openCarac_caracSetModel  $theCarac "../../foundryModels/models.lib"
openCarac_caracAddCorner $theCarac "BEST"
openCarac_caracAddCorner $theCarac "WORST"

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

foreach theRunning $theRunningList {
    # access informations:
    set theMainFile     [openCarac_runningGetMainFilePath $theRunning]
    set theIncludedFile [openCarac_runningGetSimulationIncludedFilePath $theRunning]

    puts "In the temporary folder:"
    puts "- the main file is: $theMainFile"
    puts "- the included file is: $theIncludedFile"

    openCarac_runningCreateTemporaryFolder $theRunning
}
```

4.6.2.14 openCarac_runningGetSimulatorFilesSavingFolderPath *theRunning*

Returns the value of "simulator files saving folder path" attribute of the openCarac *running*.

After calling the simulator for a specific openCarac *running* in a temporary folder created for this purpose, the result files can be copied (if openCarac *application* "simulator files copy" boolean is activated) by openCarac_runningParseSimulatorFiles from the temporary folder into a saving folder in the openCarac *configuration* directory. The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. The "simulator files saving folder path" is the destination folder for saving the found files in the temporary folder, created by openCarac↩_runningCreateTemporaryFolder, having their extension matching one of the "save filter" of the selected simulator. See access function for attribute "save filter" of the selected simulator for more informations (such as openCarac_ngspiceGet↩SaveFilter for ngspice). The value of the selected simulator can be accessed through openCarac_caracGetSimulator.

Parameters

theRunning	: openCarac *running*.

Returns

String ; openCarac *running* absolute "simulator files saving folder path" ; integer -1 if an error occurred.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the model file must exist:
openCarac_caracSetModel  $theCarac "../../foundryModels/models.lib"
openCarac_caracAddCorner $theCarac "BEST"
openCarac_caracAddCorner $theCarac "WORST"

# selection of the simulator to define the files parser:
openCarac_caracSetSimulator $theCarac "ngspice"

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

foreach theRunning $theRunningList {
    openCarac_runningCreateTemporaryFolder $theRunning
    openCarac_runningExecuteSimulator      $theRunning
    openCarac_runningParseSimulatorFiles   $theRunning

    # parsing the files also saves the output files:
    set theFolder [openCarac_runningGetSimulatorFilesSavingFolderPath $theRunning]
    puts "Simulator ouput files are saved here: $theFolder"

    openCarac_runningDeleteTemporaryFolder $theRunning
}
```

4.6.2.15 openCarac_runningParseSimulatorFiles *theRunning*

Calls the openCarac simulator files parser for the selected simulator.

openCarac comes with simulator output files parsers for each simulator it is natively compatible with. Function of the selected simulator, the appropriate files parser is called and measures are added to the openCarac *running*. This has the same effect as calling openCarac_runningSetMeasure for each measure found during the parsing. Also, depending on the parent openCarac *simulation* "extractop" attribute and the parent openCarac *carac* "extractop filter list" and "checkop list" attributes, informations at operating point are used as they were measures. It copies, if openCarac *application* "simulator files copy" boolean is activated, from the temporary folder into the saving folder in the openCarac *configuration* directory. The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGet↵SimulatorFilesCopy. The destination folder path can be accessed through openCarac_runningGetSimulatorFilesSaving↵FolderPath. The "simulator files saving folder path" is the destination folder for saving the found files in the temporary folder, created by openCarac_runningCreateTemporaryFolder, having their extension matching one of the "save filter" of the selected simulator. See access function for attribute "save filter" of the selected simulator for more informations (such as openCarac_ngspiceGetSaveFilter for ngspice). The value of the selected simulator can be accessed through openCarac_caracGetSimulator. When calling this function, before executing its main code, openCarac hook openCarac↵Hook_ON_PRE_RUNNING_PARSE_SIMULATOR_FILES is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_RUNNING_PARSE_SIMULATOR_FILES is executed.

Parameters

theRunning	: openCarac *running*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# the model file must exist:
openCarac_caracSetModel  $theCarac "../../foundryModels/models.lib"
openCarac_caracAddCorner $theCarac "BEST"
openCarac_caracAddCorner $theCarac "WORST"

# selection of the simulator to define the files parser:
openCarac_caracSetSimulator $theCarac "ngspice"

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

foreach theRunning $theRunningList {
    openCarac_runningCreateTemporaryFolder $theRunning
    openCarac_runningExecuteSimulator      $theRunning
    # call the appropriate simulator files parser:
    openCarac_runningParseSimulatorFiles   $theRunning
    openCarac_runningDeleteTemporaryFolder $theRunning
}
```

4.6.2.16 openCarac_runningSaveResults *theRunning*

Write the values of the openCarac *running* measures into a file to be used for results extraction.

For each measure that have been set to the openCarac *running*, save their step, name and value into a file. This allows to perform a results extraction even if openCarac has exit and simulator has not been executed again. Saving the results is mandatory before performing a results extraction through openCarac_runningExtractResults ; otherwise, previously set measure values are not taken into account. Measures are set to the openCarac *running* through openCarac_runningSet↩Measure. When calling this function, before executing its main code, openCarac hook openCaracHook_ON_PRE_RUN↩NING_SAVE_RESULTS is executed ; after executing its main code, openCarac hook openCaracHook_ON_POST_RU↩NNING_SAVE_RESULTS is executed.

Parameters

theRunning	: openCarac *running*.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# extract the results:
foreach theRunning $theRunningList {

    openCarac_runningSetMeasure $theRunning "step 1" "MY_MEASURE"    42
    openCarac_runningSetMeasure $theRunning "step 1" "OTHER_MEASURE" 16

    # save and extract the results:
    openCarac_runningSaveResults    $theRunning
    openCarac_runningExtractResults $theRunning

    # two structures are available:
    set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
    puts "The number of structures is: [llength $theStructureList]"
}
```

4.6.2.17 openCarac_runningSetMeasure *theRunning theResultStep theResultName theResultValue*

Sets the value of "measure" attribute of the openCarac *running*.

If a measure with the same step and the same name is already present in the openCarac *running*, it overwrites value ; otherwise it adds a new measure. Equality check of step and name values is not case-sensitive. Its name, step and value must be strings ; if its value is a double different from "NaN", it will be used by openCarac to define minimum or maximum values and to perform the checks. Tabulations are automatically substituted into spaces. Measures can be unset from the openCarac *running* through openCarac_runningUnsetMeasure.

Parameters

theRunning	: openCarac *running*.
theResultStep	: String ; measure step.
theResultName	: String ; measure name.
theResultValue	: String ; measure value ; it must be a double to be checked by openCarac.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# extract the results:
foreach theRunning $theRunningList {

    openCarac_runningSetMeasure $theRunning "step 1" "MY_MEASURE"    42
    openCarac_runningSetMeasure $theRunning "step 1" "OTHER_MEASURE" 16

    # extract the results:
    openCarac_runningSaveResults    $theRunning
    openCarac_runningExtractResults $theRunning

    # two structures are available:
    set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
    puts "The number of structures is: [llength $theStructureList]"
}
```

4.6.2.18 openCarac_runningUnsetMeasure *theRunning theResultStep theResultName*

Sets the value of "measure" attribute of the openCarac *running*.

Unsets the name, step and value for a measure to an openCarac *running*. A measure with the same step and the same name must already be present in the openCarac *running*. Equality check of step and name values is not case-sensitive. Measures can be set to the openCarac *running* through openCarac_runningSetMeasure.

Parameters

theRunning	: openCarac *running*.
theResultStep	: String ; measure step.
theResultName	: String ; measure name.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# creation of an empty configuration:
set theConf  [openCarac_configurationCreate "openCarac.conf"]
# creation of an empty carac:
set theCarac [openCarac_caracCreate $theConf "myCaracName"]
# create a simulation (file transient.inc must exist):
openCarac_applicationSetSimulationFileExtension ".inc"
set theSimu  [openCarac_simulationCreate $theCarac "transient"]

```

```
# create runnings:
set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

# extract the results:
foreach theRunning $theRunningList {

    openCarac_runningSetMeasure $theRunning "step 1" "MY_MEASURE"    42
    openCarac_runningSetMeasure $theRunning "step 1" "OTHER_MEASURE" 16

    # remove one measure:
    openCarac_runningUnsetMeasure $theRunning "step 1" "OTHER_MEASURE"

    # extract the results:
    openCarac_runningSaveResults    $theRunning
    openCarac_runningExtractResults $theRunning

    # only one structure is available:
    set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
    puts "The number of structures is: [llength $theStructureList]"
}
```

4.7 Resultstructure class

Definition of functions to manipulate openCarac *resultstructures*.

Functions

- openCarac_resultstructureGetName theResultstructure

    Returns the value of "name" attribute of the openCarac resultstructure.
- openCarac_resultstructureGetIsToBeChecked theResultstructure

    Returns the value of "is to be checked" attribute of the openCarac resultstructure.
- openCarac_resultstructureGetOneStepResultsList theResultstructure

    Returns the value of "one step results list" attribute of the openCarac resultstructure.
- openCarac_resultstructureGetVariousStepsResultsList theResultstructure

    Returns the value of "various steps results list" attribute of the openCarac resultstructure.
- openCarac_resultstructureGetMinResult theResultstructure

    Returns the value of "min result" attribute of the openCarac resultstructure.
- openCarac_resultstructureGetMaxResult theResultstructure

    Returns the value of "max result" attribute of the openCarac resultstructure.

4.7.1 Detailed Description

Definition of functions to manipulate openCarac *resultstructures*.

Here are defined every API functions that are used to access openCarac *resultstructures*. Various openCarac *resultstructures* can be defined per openCarac *running*. An openCarac *resultstructure* contains two lists of openCarac *results*, either they have only one step or various steps in the parent openCarac *running*. Their values are correct after a result extraction, i.e. calling openCarac_runningExtractResults. Each openCarac *resultstructure* may have various openCarac *result* children.

4.7.2 Function Documentation

4.7.2.1 openCarac_resultstructureGetIsToBeChecked *theResultstructure*

Returns the value of "is to be checked" attribute of the openCarac *resultstructure*.

Its value is a summary of the "is to be checked" attribute of every openCarac *result* children. If at least one openCarac *result* child is to be checked, then the openCarac *resultstructure* is considered as to be checked. Attribute "is to be checked" of openCarac *result* children can be accessed through openCarac_resultGetIsToBeChecked.

Parameters

the↩ Resultstructure	: openCarac *resultstructure*

Returns

Boolean ; 0 if the openCarac *resultstructure* is not to be checked, 1 if it is ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                if { [openCarac_resultstructureGetIsToBeChecked $theStructure] } {
                    set theName [openCarac_resultstructureGetName $theStructure]
                    puts "A result structure is to be checked: $theName"
                }

            }
        }
    }
}
```

4.7.2.2 openCarac_resultstructureGetMaxResult *theResultstructure*

Returns the value of "max result" attribute of the openCarac *resultstructure*.

In an openCarac *resultstructure*, openCarac *result* children are separated in two lists: either there is only one value of the same result in the parent openCarac *running* or there are various. This function returns a single openCarac *result* children, whatever the list it is in : the one having the maximum value in the openCarac *resultstructure*. The openCarac *result* value can be accessed through openCarac_resultGetValue. Note that the value of the openCarac *result* must be a double to be identified as the maximum value, i.e. when function openCarac_resultGetIsNotANumber returns 0.

Parameters

the←Resultstructure	: openCarac *resultstructure*

Returns

openCarac *result* ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the maximum value:
                set theResult [openCarac_resultstructureGetMaxResult $theStructure]
                set theValue  [openCarac_resultGetValue $theResult]
                puts "The maximum value in the structure is: $theValue"
            }
        }
    }
}
```

4.7.2.3 openCarac_resultstructureGetMinResult *theResultstructure*

Returns the value of "min result" attribute of the openCarac *resultstructure*.

In an openCarac *resultstructure*, openCarac *result* children are separated in two lists: either there is only one value of the same result in the parent openCarac *running* or there are various. This function returns a single openCarac *result* children, whatever the list it is in : the one having the minimum value in the openCarac *resultstructure*. The openCarac *result* value can be accessed through openCarac_resultGetValue. Note that the value of the openCarac *result* must be a double to be identified as the minimum value, i.e. when function openCarac_resultGetIsNotANumber returns 0.

Parameters

the↩ Resultstructure	: openCarac *resultstructure*

Returns

openCarac *result* ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the minimum value:
                set theResult [openCarac_resultstructureGetMinResult $theStructure]
                set theValue  [openCarac_resultGetValue $theResult]
                puts "The minimum value in the structure is: $theValue"
            }
        }
    }
}
```

4.7.2.4 openCarac_resultstructureGetName *theResultstructure*

Returns the value of "name" attribute of the openCarac *resultstructure*.

This "name" is identical to the "name" attribute of every openCarac *result* children. Name matching is not case sensitive. Attribute "name" of openCarac *result* children can be accessed through openCarac_resultGetName.

Parameters

the↩ Resultstructure	: openCarac *resultstructure*

Returns

String ; openCarac *resultstructure* name ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {
                set theName [openCarac_resultstructureGetName $theStructure]
                puts "A result structure has been extracted: $theName"
            }
        }
    }
}
```

4.7.2.5 openCarac_resultstructureGetOneStepResultsList *theResultstructure*

Returns the value of "one step results list" attribute of the openCarac *resultstructure*.

In an openCarac *resultstructure*, openCarac *result* children are separated in two lists: either there is only one value of the same result in the parent openCarac *running* or there are various. The openCarac *result* children having only one value per openCarac *running* are listed in the "one step results list". The openCarac *result* children having various values per openCarac *running* are listed in the "various steps results list". The value of "various steps results list" attribute can be accessed through openCarac_resultstructureGetVariousStepsResultsList.

Parameters

the↩ Resultstructure	: openCarac *resultstructure*

Returns

List ; openCarac *results* ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    puts "This result has only one step: $theResult"
                }

                set theResultList [openCarac_resultstructureGetVariousStepsResultsList $theStructure]
                foreach theResult $theResultList {
                    puts "This result has various steps: $theResult"
                }

            }
        }
    }
}
```

4.7.2.6 openCarac_resultstructureGetVariousStepsResultsList *theResultstructure*

Returns the value of "various steps results list" attribute of the openCarac *resultstructure*.

In an openCarac *resultstructure*, openCarac *result* children are separated in two lists: either there is only one value of the same result in the parent openCarac *running* or there are various. The openCarac *result* children having only one value per openCarac *running* are listed in the "one step results list". The openCarac *result* children having various values per openCarac *running* are listed in the "various steps results list". The value of "one step results list" attribute can be accessed through openCarac_resultstructureGetOneStepResultsList.

Parameters

the↩ Resultstructure	: openCarac *resultstructure*

Returns

List ; openCarac *results* ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    puts "This result has only one step: $theResult"
                }

                set theResultList [openCarac_resultstructureGetVariousStepsResultsList $theStructure]
                foreach theResult $theResultList {
                    puts "This result has various steps: $theResult"
                }

            }
        }
    }
}
```

4.8 Result class

Definition of functions to manipulate openCarac *results*.

Functions

- openCarac_resultGetName theResult

  Returns the value of "name" attribute of the openCarac result.
- openCarac_resultGetStep theResult

  Returns the value of "step" attribute of the openCarac result.
- openCarac_resultGetValue theResult

  Returns the value of "value" attribute of the openCarac result.
- openCarac_resultGetSimulator theResult

  Returns the value of "simulator" attribute of the openCarac result.
- openCarac_resultGetNetlist theResult

  Returns the value of "netlist" attribute of the openCarac result.
- openCarac_resultGetCorner theResult

  Returns the value of "corner" attribute of the openCarac result.
- openCarac_resultGetParam theResult

  Returns the value of "param" attribute of the openCarac result.
- openCarac_resultGetIsToBeChecked theResult

  Returns the value of "is to be checked" attribute of the openCarac result.
- openCarac_resultGetIsNotANumber theResult

  Returns the value of "is not a number" attribute of the openCarac result.
- openCarac_resultGetIsCheckOk theResult

  Returns the value of "is check OK" attribute of the openCarac result.
- openCarac_resultGetCheckMin theResult

  Returns the value of "check min" attribute of the openCarac result.
- openCarac_resultGetCheckMax theResult

  Returns the value of "check max" attribute of the openCarac result.

4.8.1 Detailed Description

Definition of functions to manipulate openCarac *results*.

Here are defined every API functions that are used to access openCarac *results*. Various openCarac *results* can be defined per openCarac *resultstructures*. An openCarac *result* contains information about measures set to an openCarac *running*, i.e. by calling openCarac_runningSetMeasure. Their values are correct after a result extraction, i.e. calling openCarac_↩ runningExtractResults.

4.8.2 Function Documentation

4.8.2.1 openCarac_resultGetCheckMax *theResult*

Returns the value of "check max" attribute of the openCarac *result*.

When extracting the results on an openCarac *running*, if a measure name matches a checkmeas or a checkop of the parent openCarac *carac*, it results to the creation of an openCarac *result* that is to be checked. This "check max" attribute is the maximum value of the checkmeas or checkop that the measure matches. If the result is not "to be checked", "check max" value is set to "NaN". The value of "to be checked" attribute of the openCarac *result* can be accessed through openCarac↩_resultGetIsToBeChecked. Results extraction is performed by openCarac_runningExtractResults. Measures can be set to the openCarac *running* through openCarac_runningSetMeasure or openCarac_runningParseSimulatorFiles. Measure values are checked function of the parent openCarac *carac* checkmeas list and checkop list, they can be accessed through openCarac_caracGetCheckmeasList and openCarac_caracGetCheckopList.

Parameters

theResult	: openCarac *result*

Returns

Double ; "check max" value ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theMin   [openCarac_resultGetCheckMin $theResult]
                    set theMax   [openCarac_resultGetCheckMax $theResult]
                    set theValue [openCarac_resultGetValue    $theResult]
                    puts "Check that: $theMin < $theValue < $theMax"
                }

            }
        }
    }
}
```

4.8.2.2 openCarac_resultGetCheckMin *theResult*

Returns the value of "check min" attribute of the openCarac *result*.

When extracting the results on an openCarac *running*, if a measure name matches a checkmeas or a checkop of the parent openCarac *carac*, it results to the creation of an openCarac *result* that is to be checked. This "check min" attribute is the minimum value of the checkmeas or checkop that the measure matches. If the result is not "to be checked", "check min" value is set to "NaN". The value of "to be checked" attribute of the openCarac *result* can be accessed through openCarac↩_resultGetIsToBeChecked. Results extraction is performed by openCarac_runningExtractResults. Measures can be set to the openCarac *running* through openCarac_runningSetMeasure or openCarac_runningParseSimulatorFiles. Measure values are checked function of the parent openCarac *carac* checkmeas list and checkop list, they can be accessed through openCarac_caracGetCheckmeasList and openCarac_caracGetCheckopList.

Parameters

theResult	: openCarac *result*

Returns

Double ; "check min" value ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theMin   [openCarac_resultGetCheckMin $theResult]
                    set theMax   [openCarac_resultGetCheckMax $theResult]
                    set theValue [openCarac_resultGetValue    $theResult]
                    puts "Check that: $theMin < $theValue < $theMax"
                }

            }
        }
    }
}
```

4.8.2.3 openCarac_resultGetCorner *theResult*

Returns the value of "corner" attribute of the openCarac *result*.

The "corner" attribute of the openCarac *result* is identical to the corner value of the parent openCarac *running*. Corner value of the parent openCarac *running* can be accessed through openCarac_runningGetCorner.

Parameters

theResult	: openCarac *result*

Returns

String ; "corner" value ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theName   [openCarac_resultGetName   $theResult]
                    set theCorner [openCarac_resultGetCorner $theResult]

                    puts "The result $theName is from a simulation of corner: $theCorner"
                }

            }
        }
    }
}
```

4.8.2.4 openCarac_resultGetIsCheckOk *theResult*

Returns the value of "is check OK" attribute of the openCarac *result*.

When extracting the results on an openCarac *running*, if a measure name matches a checkmeas or a checkop of the parent openCarac *carac*, it results to the creation of an openCarac *result* that is to be checked. It is considered as OK when its value is superior or equal to the "check min" value and inferior or equal to the "check max" value. When its value is not a double (i.e. when function openCarac_resultGetIsNotANumber returns 1), the result is not considered as OK. The openCarac *result* value can be accessed through openCarac_resultGetValue. The openCarac *result* "check min" and "check max" values can be accessed through openCarac_resultGetCheckMin and openCarac_resultGetCheckMax. Results extraction is performed by openCarac_runningExtractResults. Measures can be set to the openCarac *running* through openCarac_runningSetMeasure or openCarac_runningParseSimulatorFiles. Measure values are checked function of the parent openCarac *carac* checkmeas list and checkop list, they can be accessed through openCarac_caracGet↩ CheckmeasList and openCarac_caracGetCheckopList.

Parameters

theResult	: openCarac *result*

Returns

Boolean ; 0 if the openCarac *result* is not considered OK, 1 if it is ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theName [openCarac_resultGetName $theResult]

                    if { [openCarac_resultGetIsNotANumber $theResult] } {
                        puts "Result $theName is not a number."
                    } elseif { [openCarac_resultGetIsToBeChecked $theResult] } {
                        if { [openCarac_resultGetIsCheckOk $theResult] } {
                            puts "Result $theName is OK."
                        } else {
                            puts "Result $theName is not OK."
                        }
                    }

                }

            }
        }
    }
}
```

4.8.2.5 openCarac_resultGetIsNotANumber *theResult*

Returns the value of "is not a number" attribute of the openCarac *result*.

Checks if the openCarac *result* value is not a number. The openCarac *result* value can be accessed through openCarac↩ _resultGetValue. Measures can be set to the openCarac *running* through openCarac_runningSetMeasure or openCarac↩ _runningParseSimulatorFiles.

Parameters

theResult	: openCarac *result*

Returns

Boolean ; 0 if the openCarac *result* is a double, 1 if it is not ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theName [openCarac_resultGetName $theResult]

                    if { [openCarac_resultGetIsNotANumber $theResult] } {
                        puts "Result $theName is not a number."
                    } elseif { [openCarac_resultGetIsToBeChecked $theResult] } {
                        if { [openCarac_resultGetIsCheckOk $theResult] } {
                            puts "Result $theName is OK."
                        } else {
                            puts "Result $theName is not OK."
                        }
                    }

                }

            }
        }
    }
}
```

4.8.2.6 openCarac_resultGetIsToBeChecked *theResult*

Returns the value of "is to be checked" attribute of the openCarac *result*.

When extracting the results on an openCarac *running*, if a measure name matches a checkmeas or a checkop of the parent openCarac *carac*, it results to the creation of an openCarac *result* that is to be checked. Results extraction is performed by openCarac_runningExtractResults. Measures can be set to the openCarac *running* through openCarac_runningSet↩ Measure or openCarac_runningParseSimulatorFiles. Measure values are checked function of the parent openCarac *carac* checkmeas list and checkop list, they can be accessed through openCarac_caracGetCheckmeasList and openCarac_↩ caracGetCheckopList.

Parameters

theResult	: openCarac *result*

Returns

Boolean ; 0 if the openCarac *result* is not to be checked, 1 if it is ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theName [openCarac_resultGetName $theResult]

                    if { [openCarac_resultGetIsNotANumber $theResult] } {
                        puts "Result $theName is not a number."
                    } elseif { [openCarac_resultGetIsToBeChecked $theResult] } {
                        if { [openCarac_resultGetIsCheckOk $theResult] } {
                            puts "Result $theName is OK."
                        } else {
                            puts "Result $theName is not OK."
                        }
                    }

                }

            }
        }
    }
}
```

4.8.2.7 openCarac_resultGetName *theResult*

Returns the value of "name" attribute of the openCarac *result*.

The "name" attribute of the openCarac *result* is identical to the measure name that has been set to the parent openCarac *running*. Measures can be set through openCarac_runningSetMeasure.

Parameters

theResult	: openCarac *result*

Returns

String ; openCarac *result* name ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theName [openCarac_resultGetName $theResult]

                    if { [openCarac_resultGetIsNotANumber $theResult] } {
                        puts "Result $theName is not a number."
                    } elseif { [openCarac_resultGetIsToBeChecked $theResult] } {
                        if { [openCarac_resultGetIsCheckOk $theResult] } {
                            puts "Result $theName is OK."
                        } else {
                            puts "Result $theName is not OK."
                        }
                    }

                }

            }
        }
    }
}
```

4.8.2.8 openCarac_resultGetNetlist *theResult*

Returns the value of "netlist" attribute of the openCarac *result*.

The "netlist" attribute of the openCarac *result* is identical to the netlist value of the parent openCarac *running*. Netlist value of the parent openCarac *running* can be accessed through openCarac_runningGetNetlist.

Parameters

theResult	: openCarac *result*

Returns

String ; "netlist" value ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theName    [openCarac_resultGetName    $theResult]
                    set theNetlist [openCarac_resultGetNetlist $theResult]

                    puts "The result $theName is from a simulation of netlist: $theNetlist"
                }

            }
        }
    }
}
```

4.8.2.9 openCarac_resultGetParam *theResult*

Returns the value of "param" attribute of the openCarac *result*.

The "param" attribute of the openCarac *result* is identical to the param value of the parent openCarac *running*. Param value of the parent openCarac *running* can be accessed through openCarac_runningGetParam.

Parameters

theResult	: openCarac *result*

Returns

String ; "param" value ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theName  [openCarac_resultGetName  $theResult]
                    set theParam [openCarac_resultGetParam $theResult]

                    puts "The result $theName is from a simulation of param: $theParam"
                }

            }
        }
    }
}
```

4.8.2.10 openCarac_resultGetSimulator *theResult*

Returns the value of "simulator" attribute of the openCarac *result*.

The "simulator" attribute of the openCarac *result* is identical to the simulator value that has been set to the parent openCarac *carac*, concatenated with its version that has been detected during the simulator execution. Simulator value can be set through openCarac_caracSetSimulator. Simulator version is detected when calling openCarac_runningExecuteSimulator.

Parameters

theResult	: openCarac *result*

Returns

String ; "simulator" value ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theName      [openCarac_resultGetName      $theResult]
                    set theSimulator [openCarac_resultGetSimulator $theResult]

                    puts "The result $theName is from a simulation of: $theSimulator"
                }

            }
        }
    }
}
```

4.8.2.11 openCarac_resultGetStep *theResult*

Returns the value of "step" attribute of the openCarac *result*.

The "step" attribute of the openCarac *result* is identical to the measure step that has been set to the parent openCarac *running*. Measures can be set through openCarac_runningSetMeasure.

Parameters

theResult	: openCarac *result*

Returns

String ; "step" value ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theName [openCarac_resultGetName $theResult]
                    set theStep [openCarac_resultGetStep $theResult]

                    puts "The result $theName is at step: $theStep"
                }

            }
        }
    }
}
```

4.8.2.12 openCarac_resultGetValue *theResult*

Returns the value of "value" attribute of the openCarac *result*.

The "value" attribute of the openCarac *result* is identical to the measure value that has been set to the parent openCarac *running*. Measures can be set through openCarac_runningSetMeasure.

Parameters

theResult	: openCarac *result*

Returns

String ; value ; integer -1 if an error occurred.

Example

```
# access every carac of every configuration:
foreach theConfiguration [openCarac_applicationGetLoadedConfigurationsList] {
    foreach theCarac [openCarac_configurationGetCaracsList $theConfiguration] {
        # create runnings:
        set theRunningList [openCarac_caracMakeReadyForRunnings $theCarac]

        foreach theRunning $theRunningList {
            # extract the results:
            openCarac_runningExtractResults $theRunning
            # the structures are available:
            set theStructureList [openCarac_runningGetResultstructuresList $theRunning]
            foreach theStructure $theStructureList {

                # access the results:
                set theResultList [openCarac_resultstructureGetOneStepResultsList $theStructure]
                foreach theResult $theResultList {
                    set theMin   [openCarac_resultGetCheckMin $theResult]
                    set theMax   [openCarac_resultGetCheckMax $theResult]
                    set theValue [openCarac_resultGetValue    $theResult]
                    puts "Check that: $theMin < $theValue < $theMax"
                }

            }
        }
    }
}
```

4.9 Ngspice simulator

Definition of functions to interact with openCarac settings for *ngspice* simulator.

Functions

- openCarac_ngspiceGetCommand

  Returns the value of "command" attribute for ngspice simulator.
- openCarac_ngspiceSetCommand value

  Sets the value of "command" attribute for ngspice simulator.
- openCarac_ngspiceGetCheckOptions

  Returns the value of "check options" attribute for ngspice simulator.
- openCarac_ngspiceSetCheckOptions value

  Sets the value of "check options" attribute for ngspice simulator.
- openCarac_ngspiceGetRunOptions

  Returns the value of "run options" attribute for ngspice simulator.
- openCarac_ngspiceSetRunOptions value

  Sets the value of "run options" attribute for ngspice simulator.
- openCarac_ngspiceGetLogExtension

  Returns the value of "log extension" attribute for ngspice simulator.
- openCarac_ngspiceSetLogExtension value

  Sets the value of "log extension" attribute for ngspice simulator.
- openCarac_ngspiceGetSaveFilter

  Returns the value of "save filter" attribute for ngspice simulator.
- openCarac_ngspiceSetSaveFilter value

  Sets the value of "save filter" attribute for ngspice simulator.
- openCarac_ngspiceGetToRemoveInCheckMode

  Returns the value of "to remove in check mode" attribute for ngspice simulator.
- openCarac_ngspiceSetToRemoveInCheckMode value

  Sets the value of "to remove in check mode" attribute for ngspice simulator.
- openCarac_ngspiceGetCommentSyntax

  Returns the value of "comment syntax" attribute for ngspice simulator.
- openCarac_ngspiceSetCommentSyntax value

  Sets the value of "comment syntax" attribute for ngspice simulator.
- openCarac_ngspiceGetIncDirective

  Returns the value of "inc directive" attribute for ngspice simulator.
- openCarac_ngspiceSetIncDirective value

  Sets the value of "inc directive" attribute for ngspice simulator.
- openCarac_ngspiceGetLibDirective

  Returns the value of "lib directive" attribute for ngspice simulator.
- openCarac_ngspiceSetLibDirective value

  Sets the value of "lib directive" attribute for ngspice simulator.
- openCarac_ngspiceGetParamDirective

  Returns the value of "param directive" attribute for ngspice simulator.
- openCarac_ngspiceSetParamDirective value

  Sets the value of "param directive" attribute for ngspice simulator.
- openCarac_ngspiceGetParamEquality

  Returns the value of "param equality" attribute for ngspice simulator.

- openCarac_ngspiceSetParamEquality value

  Sets the value of "param equality" attribute for ngspice simulator.
- openCarac_ngspiceGetStringDelimiter

  Returns the value of "string delimiter" attribute for ngspice simulator.
- openCarac_ngspiceSetStringDelimiter value

  Sets the value of "string delimiter" attribute for ngspice simulator.
- openCarac_ngspiceActivateCaseSensitivity

  Sets openCarac ngspice boolean "case sensitivity" to "1".
- openCarac_ngspiceDeactivateCaseSensitivity

  Sets openCarac ngspice boolean "case sensitivity" to "0".
- openCarac_ngspiceGetCaseSensitivity

  Returns the value of "case sensitivity" attribute for ngspice simulator.
- openCarac_ngspiceActivateDirectoryChange

  Sets openCarac ngspice boolean "directory change" to "1".
- openCarac_ngspiceDeactivateDirectoryChange

  Sets openCarac ngspice boolean "directory change" to "0".
- openCarac_ngspiceGetDirectoryChange

  Returns the value of "directory change" attribute for ngspice simulator.
- openCarac_ngspiceActivateMonitorErrorCode

  Sets openCarac ngspice boolean "monitor error code" to "1".
- openCarac_ngspiceDeactivateMonitorErrorCode

  Sets openCarac ngspice boolean "monitor error code" to "0".
- openCarac_ngspiceGetMonitorErrorCode

  Returns the value of "monitor error code" attribute for ngspice simulator.

4.9.1 Detailed Description

Definition of functions to interact with openCarac settings for *ngspice* simulator.

openCarac aims to be compatible with various spice simulators. Since different simulators have different syntax and no Spice parser is available in openCarac, a configuration must be done for openCarac to execute it properly. In this module are described every function used to make openCarac fully compatible with *ngspice* simulator.

4.9.2 Function Documentation

4.9.2.1 openCarac_ngspiceActivateCaseSensitivity

Sets openCarac ngspice boolean "case sensitivity" to "1".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac ngspice files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop↩
FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_ngspiceGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_ngspiceActivateCaseSensitivity

# verify its new value:
if { [openCarac_ngspiceGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.9.2.2 openCarac_ngspiceActivateDirectoryChange

Sets openCarac ngspice boolean "directory change" to "1".

When executing *ngspice* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *ngspice* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_ngspiceGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_ngspiceActivateDirectoryChange

# verify its new value:
if { [openCarac_ngspiceGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.9.2.3 openCarac_ngspiceActivateMonitorErrorCode

Sets openCarac ngspice boolean "monitor error code" to "1".

When executing *ngspice* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *ngspice* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_ngspiceGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_ngspiceActivateMonitorErrorCode

# verify its new value:
if { [openCarac_ngspiceGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.9.2.4 openCarac_ngspiceDeactivateCaseSensitivity

Sets openCarac ngspice boolean "case sensitivity" to "0".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac ngspice files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop↩FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_ngspiceGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_ngspiceDeactivateCaseSensitivity

# verify its new value:
if { [openCarac_ngspiceGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.9.2.5 openCarac_ngspiceDeactivateDirectoryChange

Sets openCarac ngspice boolean "directory change" to "0".

When executing *ngspice* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *ngspice* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_ngspiceGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_ngspiceDeactivateDirectoryChange

# verify its new value:
if { [openCarac_ngspiceGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.9.2.6 openCarac_ngspiceDeactivateMonitorErrorCode

Sets openCarac ngspice boolean "monitor error code" to "0".

When executing *ngspice* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *ngspice* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_ngspiceGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_ngspiceDeactivateMonitorErrorCode

# verify its new value:
if { [openCarac_ngspiceGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.9.2.7 openCarac_ngspiceGetCaseSensitivity

Returns the value of "case sensitivity" attribute for *ngspice* simulator.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac ngspice files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractopFilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be set through openCarac_ngspiceActivateCaseSensitivity and openCarac_ngspiceDeactivateCaseSensitivity.

Returns

Boolean ; 0 if "case sensitivity" attribute of *ngspice* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_ngspiceActivateCaseSensitivity

# verify its new value:
if { [openCarac_ngspiceGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.9.2.8 openCarac_ngspiceGetCheckOptions

Returns the value of "check options" attribute for *ngspice* simulator.

The *ngspice* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_ngspiceSetCheckOptions.

Returns

String ; *ngspice* command check options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_ngspiceSetCommand      "/usr/bin/ngspice"
openCarac_ngspiceSetCheckOptions "-b -n"
openCarac_ngspiceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_ngspiceGetCheckOptions]
} else {
    set theOptions [openCarac_ngspiceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_ngspiceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.9.2.9 openCarac_ngspiceGetCommand

Returns the value of "command" attribute for *ngspice* simulator.

This returns the command to execute *ngspice* simulator. This command is executed by openCarac through the TC↩L exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of open↩Carac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac_ngspiceGetCheckOptions and openCarac_ngspiceGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be changed through openCarac_ngspiceGetCommand.

Returns

String ; *ngspice* command ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_ngspiceSetCommand      "/usr/bin/ngspice"
openCarac_ngspiceSetCheckOptions "-b -n"
openCarac_ngspiceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_ngspiceGetCheckOptions]
} else {
    set theOptions [openCarac_ngspiceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_ngspiceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.9.2.10 openCarac_ngspiceGetCommentSyntax

Returns the value of "comment syntax" attribute for *ngspice* simulator.

Its value is a non-empty string that is not a list. When creating a temporary folder through openCarac_runningCreate↩TemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be set through openCarac_ngspiceSetCommentSyntax.

Returns

String ; comment syntax, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the comment syntax:
openCarac_ngspiceSetCommentSyntax "**"

set theComment "[openCarac_ngspiceGetCommentSyntax] This is a comment."
puts $theComment
```

4.9.2.11 openCarac_ngspiceGetDirectoryChange

Returns the value of "directory change" attribute for *ngspice* simulator.

When executing *ngspice* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *ngspice* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, open↩ Carac changes back the previous location. Its value can be set through openCarac_ngspiceActivateDirectoryChange and openCarac_ngspiceDeactivateDirectoryChange.

Returns

Boolean ; 0 if "directory change" attribute of *ngspice* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_ngspiceActivateDirectoryChange

# verify its new value:
if { [openCarac_ngspiceGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.9.2.12 openCarac_ngspiceGetIncDirective

Returns the value of "inc directive" attribute for *ngspice* simulator.

Its value is a non-empty string that is not a list ; also, it is different from the ngspice "lib directive" and ngspice "param directive". To define it, case sensitivity depends on the ngspice "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac ngspice "lib directive" and "param directive" can be accessed through openCarac_ngspiceGet↩ LibDirective and openCarac_ngspiceGetParamDirective. The value of openCarac ngspice "case sensitivity" boolean can be accessed through openCarac_ngspiceGetCaseSensitivity. Its value can be set through openCarac_ngspiceSetInc↩ Directive.

Returns

String ; inc directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_ngspiceSetIncDirective    ".INCLUDE"
openCarac_ngspiceSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_ngspiceGetIncDirective]
set theDelim     [openCarac_ngspiceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.9.2.13 openCarac_ngspiceGetLibDirective

Returns the value of "lib directive" attribute for *ngspice* simulator.

Its value is a non-empty string that is not a list ; also, it is different from the ngspice "inc directive" and ngspice "param directive". To define it, case sensitivity depends on the ngspice "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of openCarac ngspice "inc directive" and "param directive" can be accessed through openCarac_ngspiceGetIncDirective and openCarac_ngspice←GetParamDirective. The value of openCarac ngspice "case sensitivity" boolean can be accessed through openCarac_←ngspiceGetCaseSensitivity. Its value can be set through openCarac_ngspiceSetLibDirective.

Returns

String ; lib directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_ngspiceSetLibDirective    ".LIB"
openCarac_ngspiceSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_ngspiceGetLibDirective]
set theDelim     [openCarac_ngspiceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.9.2.14 openCarac_ngspiceGetLogExtension

Returns the value of "log extension" attribute for *ngspice* simulator.

This returns the log file extension to print what is returned by the *ngspice* command in. The command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension is a lower case non-empty string, not a list itself, of at least two characters and starting with a dot (.). The value of the *ngspice* command can be accessed through openCarac_ngspiceGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be set through openCarac_ngspiceSetLogExtension.

Returns

String ; log extension, single word, in lower case, of at least two characters and starting with a dot (.).

Example

```
# change the log extension:
openCarac_ngspiceSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_ngspiceGetCheckOptions]
} else {
    set theOptions [openCarac_ngspiceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_ngspiceGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_ngspiceGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.9.2.15 openCarac_ngspiceGetMonitorErrorCode

Returns the value of "monitor error code" attribute for *ngspice* simulator.

When executing *ngspice* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *ngspice* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be set through openCarac_ngspiceActivateMonitorErrorCode and openCarac_ngspiceDeactivateMonitorErrorCode.

Returns

Boolean ; 0 if "monitor error code" attribute of *ngspice* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_ngspiceActivateMonitorErrorCode

# verify its new value:
if { [openCarac_ngspiceGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.9.2.16 openCarac_ngspiceGetParamDirective

Returns the value of "param directive" attribute for *ngspice* simulator.

Its value must be a non-empty string that is not a list ; also, it is different from the ngspice "inc directive" and ngspice "lib directive". To define it, case sensitivity depends on the ngspice "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac ngspice "inc directive" and "lib directive" can be accessed through openCarac_ngspiceGetIncDirective and openCarac_ngspiceGetLibDirective. The value of openCarac ngspice "case sensitivity" boolean can be accessed through openCarac_ngspiceGetCaseSensitivity. The value of "param equality" can be accessed through openCarac_ngspiceGetParamEquality. Its value can be set through openCarac_ngspiceSetParamDirective.

Returns

String ; param directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_ngspiceSetParamDirective ".PARAM"
openCarac_ngspiceSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_ngspiceGetParamDirective]
set theEqual     [openCarac_ngspiceGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.9.2.17 openCarac_ngspiceGetParamEquality

Returns the value of "param equality" attribute for *ngspice* simulator.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac ngspice "param directive" can be accessed through openCarac_ngspiceGetParamDirective. Its value can be set through openCarac_ngspiceSetParamEquality.

Returns

String ; param equality ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_ngspiceSetParamDirective ".PARAM"
openCarac_ngspiceSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_ngspiceGetParamDirective]
set theEqual     [openCarac_ngspiceGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.9.2.18 openCarac_ngspiceGetRunOptions

Returns the value of "run options" attribute for *ngspice* simulator.

The *ngspice* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_ngspiceSetRunOptions.

Returns

String ; *ngspice* command run options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_ngspiceSetCommand      "/usr/bin/ngspice"
openCarac_ngspiceSetCheckOptions "-b -n"
openCarac_ngspiceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_ngspiceGetCheckOptions]
} else {
    set theOptions [openCarac_ngspiceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_ngspiceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.9.2.19 openCarac_ngspiceGetSaveFilter

Returns the value of "save filter" attribute for *ngspice* simulator.

When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac↩_runningGetSimulatorFilesSavingFolderPath. Save filter is a list of strings in lower case, each of them being a single word starting with a dot (.). The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be set through openCarac_ngspiceSetSaveFilter.

Returns

List ; strings in lower case, single words starting with a dot ; integer -1 if an error occurred.

Example

```
set theExtensionsList [openCarac_ngspiceGetSaveFilter]

# define which files are not saved by openCarac:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*"] {

    set theExtension [string tolower [file extension $theFile]]

    if { [lsearch $theExtensionsList $theExtension] == -1 } {
        openCarac_warning "This file will not be saved by openCarac: $theFile"
    }

}
```

4.9.2.20 openCarac_ngspiceGetStringDelimiter

Returns the value of "string delimiter" attribute for *ngspice* simulator.

Its value is an empty string or a single character. When creating a temporary folder through openCarac_runningCreate↩TemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be set through openCarac_ngspice↩SetStringDelimiter.

Returns

String ; string delimiter, empty string or single character ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_ngspiceSetIncDirective    ".INCLUDE"
openCarac_ngspiceSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_ngspiceGetIncDirective]
set theDelim     [openCarac_ngspiceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.9.2.21 openCarac_ngspiceGetToRemoveInCheckMode

Returns the value of "to remove in check mode" attribute for *ngspice* simulator.

This is a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and ngspice comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the ngspice "case sensitivity" attribute (accessible through openCarac_ngspiceGetCaseSensitivity). The value of open↩ Carac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_ngspiceSetToRemoveInCheckMode.

Returns

List ; Strings that are not empty and not lists themselves ; integer -1 if an error occurred.

Example

```
# set the list of directives to remove:
openCarac_ngspiceSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_ngspiceGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

4.9.2.22 openCarac_ngspiceSetCheckOptions *value*

Sets the value of "check options" attribute for *ngspice* simulator.

The *ngspice* command is executed by openCarac through the TCL exec command when calling openCarac_running↩ ExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGet↩ CustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac↩ _applicationGetCheckMode. Its value can be accessed through openCarac_ngspiceGetCheckOptions.

Parameters

value	: String ; *ngspice* command check options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_ngspiceSetCommand      "/usr/bin/ngspice"
openCarac_ngspiceSetCheckOptions "-b -n"
openCarac_ngspiceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_ngspiceGetCheckOptions]
} else {
    set theOptions [openCarac_ngspiceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_ngspiceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.9.2.23 openCarac_ngspiceSetCommand *value*

Sets the value of "command" attribute for *ngspice* simulator.

This sets the command to execute *ngspice* simulator. This command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of openCarac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac↩_ngspiceGetCheckOptions and openCarac_ngspiceGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be accessed through openCarac_ngspiceGetCommand.

Parameters

value	: String ; *ngspice* command.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_ngspiceSetCommand      "/usr/bin/ngspice"
openCarac_ngspiceSetCheckOptions "-b -n"
openCarac_ngspiceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_ngspiceGetCheckOptions]
} else {
    set theOptions [openCarac_ngspiceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_ngspiceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.9.2.24 openCarac_ngspiceSetCommentSyntax *value*

Sets the value of "comment syntax" attribute for *ngspice* simulator.

Its value must be a non-empty string that is not a list. When creating a temporary folder through openCarac_running↩ CreateTemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be accessed through openCarac_ngspiceGet↩ CommentSyntax.

Parameters

value	: String ; comment syntax, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the comment syntax:
openCarac_ngspiceSetCommentSyntax "**"

set theComment "[openCarac_ngspiceGetCommentSyntax] This is a comment."
puts $theComment
```

4.9.2.25 openCarac_ngspiceSetIncDirective *value*

Sets the value of "inc directive" attribute for *ngspice* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the ngspice "lib directive" and ngspice "param directive". To define it, case sensitivity depends on the ngspice "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac ngspice "lib directive" and "param directive" can be accessed through openCarac_ngspiceGetLibDirective and openCarac_ngspiceGetParamDirective. The value of openCarac ngspice "case sensitivity" boolean can be accessed through openCarac_ngspiceGetCaseSensitivity. Its value can be accessed through openCarac_ngspiceGetIncDirective.

Parameters

value	: String ; inc directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_ngspiceSetIncDirective    ".INCLUDE"
openCarac_ngspiceSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_ngspiceGetIncDirective]
set theDelim     [openCarac_ngspiceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.9.2.26 openCarac_ngspiceSetLibDirective *value*

Sets the value of "lib directive" attribute for *ngspice* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the ngspice "inc directive" and ngspice "param directive". To define it, case sensitivity depends on the ngspice "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of openCarac ngspice "inc directive" and "param directive" can be accessed through openCarac_ngspiceGetIncDirective and openCarac_ngspiceGetParamDirective. The value of openCarac ngspice "case sensitivity" boolean can be accessed through openCarac_ngspiceGetCaseSensitivity. Its value can be accessed through openCarac_ngspiceGetLibDirective.

Parameters

value	: String ; lib directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_ngspiceSetLibDirective    ".LIB"
openCarac_ngspiceSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_ngspiceGetLibDirective]
set theDelim     [openCarac_ngspiceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.9.2.27 openCarac_ngspiceSetLogExtension *value*

Sets the value of "log extension" attribute for *ngspice* simulator.

This sets the log file extension to print what is returned by the *ngspice* command in. The command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension must be a non-empty string, not a list itself, of at least two characters and starting with a dot (.), open↩ Carac automatically converts it to lower case. If the log file extension does not appear in the "save filter" attribute of openCarac *ngspice* simulator, accessible through openCarac_ngspiceGetSaveFilter, it is automatically added. The value of the *ngspice* command can be accessed through openCarac_ngspiceGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be accessed through openCarac_ngspiceGetLogExtension.

Parameters

value	: String ; log extension, single word, of at least two characters and starting with a dot (.).

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the log extension:
openCarac_ngspiceSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_ngspiceGetCheckOptions]
} else {
    set theOptions [openCarac_ngspiceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_ngspiceGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_ngspiceGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.9.2.28 openCarac_ngspiceSetParamDirective *value*

Sets the value of "param directive" attribute for *ngspice* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the ngspice "inc directive" and ngspice "lib directive". To define it, case sensitivity depends on the ngspice "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac ngspice "inc directive" and "lib directive" can be accessed through openCarac_ngspiceGetIncDirective and openCarac_ngspiceGetLibDirective. The value of openCarac ngspice "case sensitivity" boolean can be accessed through openCarac_ngspiceGetCaseSensitivity. The value of "param equality" can be accessed through openCarac_ngspiceGetParamEquality. Its value can be accessed through openCarac_ngspiceGetParamDirective.

Parameters

value	: String ; param directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_ngspiceSetParamDirective ".PARAM"
openCarac_ngspiceSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_ngspiceGetParamDirective]
set theEqual     [openCarac_ngspiceGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.9.2.29 openCarac_ngspiceSetParamEquality *value*

Sets the value of "param equality" attribute for *ngspice* simulator.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac ngspice "param directive" can be accessed through openCarac_ngspiceGetParamDirective. Its value can be accessed through openCarac_ngspiceGetParamEquality.

Parameters

value	: String ; param equality.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_ngspiceSetParamDirective ".PARAM"
openCarac_ngspiceSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_ngspiceGetParamDirective]
set theEqual     [openCarac_ngspiceGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.9.2.30 openCarac_ngspiceSetRunOptions *value*

Sets the value of "run options" attribute for *ngspice* simulator.

The *ngspice* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_ngspiceGetRunOptions.

Parameters

value	: String ; *ngspice* command run options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_ngspiceSetCommand      "/usr/bin/ngspice"
openCarac_ngspiceSetCheckOptions "-b -n"
openCarac_ngspiceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_ngspiceGetCheckOptions]
} else {
    set theOptions [openCarac_ngspiceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_ngspiceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.9.2.31 openCarac_ngspiceSetSaveFilter *value*

Sets the value of "save filter" attribute for *ngspice* simulator.

This must be a list of strings, each of them being a single word starting with a dot (.), openCarac automatically converts them to lower case. When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac_runningGetSimulatorFilesSavingFolderPath. The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be accessed through openCarac↩_ngspiceGetSaveFilter.

Parameters

value	: List ; strings, single words starting with a dot.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
set theFilesExtensionsFilter [openCarac_applicationGetFilesExtensionFilter]
set theSaveFilterList         [list]

# use the extensions of the files in the current directory:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*.*"] {

    set theExtension [string tolower [file extension $theFile]]

    # ignore the files that have been copied by openCarac:
    if { [lsearch $theFilesExtensionsFilter $theExtension] == -1 } {
        continue
    }

    if { [lsearch $theSaveFilterList $theExtension] == -1 } {
        lappend theSaveFilterList $theExtension
    }

}

# apply this filter to openCarac:
openCarac_ngspiceSetSaveFilter $theSaveFilterList
```

4.9.2.32 openCarac_ngspiceSetStringDelimiter *value*

Sets the value of "string delimiter" attribute for *ngspice* simulator.

Its value must be an empty string or a single character. When creating a temporary folder through openCarac_running↩CreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be accessed through openCarac_ngspiceGetStringDelimiter.

Parameters

value	: String ; string delimiter, empty string or single character.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_ngspiceSetIncDirective     ".INCLUDE"
openCarac_ngspiceSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_ngspiceGetIncDirective]
set theDelim     [openCarac_ngspiceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.9.2.33 openCarac_ngspiceSetToRemoveInCheckMode *value*

Sets the value of "to remove in check mode" attribute for *ngspice* simulator.

This must be a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and ngspice comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the ngspice "case sensitivity" attribute (accessible through openCarac_ngspiceGetCaseSensitivity). The value of open↩ Carac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_ngspiceGetToRemoveInCheckMode.

Parameters

value	: List ; Strings that are not empty and not lists themselves.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# set the list of directives to remove:
openCarac_ngspiceSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_ngspiceGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

4.10 Gnucap simulator

Definition of functions to interact with openCarac settings for *gnucap* simulator.

Functions

- openCarac_gnucapGetCommand

    Returns the value of "command" attribute for gnucap simulator.
- openCarac_gnucapSetCommand value

    Sets the value of "command" attribute for gnucap simulator.
- openCarac_gnucapGetCheckOptions

    Returns the value of "check options" attribute for gnucap simulator.
- openCarac_gnucapSetCheckOptions value

    Sets the value of "check options" attribute for gnucap simulator.
- openCarac_gnucapGetRunOptions

    Returns the value of "run options" attribute for gnucap simulator.
- openCarac_gnucapSetRunOptions value

    Sets the value of "run options" attribute for gnucap simulator.
- openCarac_gnucapGetLogExtension

    Returns the value of "log extension" attribute for gnucap simulator.
- openCarac_gnucapSetLogExtension value

    Sets the value of "log extension" attribute for gnucap simulator.
- openCarac_gnucapGetSaveFilter

    Returns the value of "save filter" attribute for gnucap simulator.
- openCarac_gnucapSetSaveFilter value

    Sets the value of "save filter" attribute for gnucap simulator.
- openCarac_gnucapGetToRemoveInCheckMode

    Returns the value of "to remove in check mode" attribute for gnucap simulator.
- openCarac_gnucapSetToRemoveInCheckMode value

    Sets the value of "to remove in check mode" attribute for gnucap simulator.
- openCarac_gnucapGetCommentSyntax

    Returns the value of "comment syntax" attribute for gnucap simulator.
- openCarac_gnucapSetCommentSyntax value

    Sets the value of "comment syntax" attribute for gnucap simulator.
- openCarac_gnucapGetIncDirective

    Returns the value of "inc directive" attribute for gnucap simulator.
- openCarac_gnucapSetIncDirective value

    Sets the value of "inc directive" attribute for gnucap simulator.
- openCarac_gnucapGetLibDirective

    Returns the value of "lib directive" attribute for gnucap simulator.
- openCarac_gnucapSetLibDirective value

    Sets the value of "lib directive" attribute for gnucap simulator.
- openCarac_gnucapGetParamDirective

    Returns the value of "param directive" attribute for gnucap simulator.
- openCarac_gnucapSetParamDirective value

    Sets the value of "param directive" attribute for gnucap simulator.
- openCarac_gnucapGetParamEquality

    Returns the value of "param equality" attribute for gnucap simulator.

- openCarac_gnucapSetParamEquality value

  Sets the value of "param equality" attribute for gnucap simulator.
- openCarac_gnucapGetStringDelimiter

  Returns the value of "string delimiter" attribute for gnucap simulator.
- openCarac_gnucapSetStringDelimiter value

  Sets the value of "string delimiter" attribute for gnucap simulator.
- openCarac_gnucapActivateCaseSensitivity

  Sets openCarac gnucap boolean "case sensitivity" to "1".
- openCarac_gnucapDeactivateCaseSensitivity

  Sets openCarac gnucap boolean "case sensitivity" to "0".
- openCarac_gnucapGetCaseSensitivity

  Returns the value of "case sensitivity" attribute for gnucap simulator.
- openCarac_gnucapActivateDirectoryChange

  Sets openCarac gnucap boolean "directory change" to "1".
- openCarac_gnucapDeactivateDirectoryChange

  Sets openCarac gnucap boolean "directory change" to "0".
- openCarac_gnucapGetDirectoryChange

  Returns the value of "directory change" attribute for gnucap simulator.
- openCarac_gnucapActivateMonitorErrorCode

  Sets openCarac gnucap boolean "monitor error code" to "1".
- openCarac_gnucapDeactivateMonitorErrorCode

  Sets openCarac gnucap boolean "monitor error code" to "0".
- openCarac_gnucapGetMonitorErrorCode

  Returns the value of "monitor error code" attribute for gnucap simulator.

4.10.1 Detailed Description

Definition of functions to interact with openCarac settings for *gnucap* simulator.

openCarac aims to be compatible with various spice simulators. Since different simulators have different syntax and no Spice parser is available in openCarac, a configuration must be done for openCarac to execute it properly. In this module are described every function used to make openCarac fully compatible with *gnucap* simulator.

4.10.2 Function Documentation

4.10.2.1 openCarac_gnucapActivateCaseSensitivity

Sets openCarac gnucap boolean "case sensitivity" to "1".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac gnucap files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop↩FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_gnucapGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_gnucapActivateCaseSensitivity

# verify its new value:
if { [openCarac_gnucapGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.10.2.2 openCarac_gnucapActivateDirectoryChange

Sets openCarac gnucap boolean "directory change" to "1".

When executing *gnucap* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *gnucap* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_gnucapGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_gnucapActivateDirectoryChange

# verify its new value:
if { [openCarac_gnucapGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.10.2.3 openCarac_gnucapActivateMonitorErrorCode

Sets openCarac gnucap boolean "monitor error code" to "1".

When executing *gnucap* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *gnucap* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_gnucapGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_gnucapActivateMonitorErrorCode

# verify its new value:
if { [openCarac_gnucapGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.10.2.4 openCarac_gnucapDeactivateCaseSensitivity

Sets openCarac gnucap boolean "case sensitivity" to "0".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac gnucap files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop↩FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_gnucapGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_gnucapDeactivateCaseSensitivity

# verify its new value:
if { [openCarac_gnucapGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.10.2.5 openCarac_gnucapDeactivateDirectoryChange

Sets openCarac gnucap boolean "directory change" to "0".

When executing *gnucap* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *gnucap* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_gnucapGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_gnucapDeactivateDirectoryChange

# verify its new value:
if { [openCarac_gnucapGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.10.2.6 openCarac_gnucapDeactivateMonitorErrorCode

Sets openCarac gnucap boolean "monitor error code" to "0".

When executing *gnucap* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *gnucap* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_gnucapGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_gnucapDeactivateMonitorErrorCode

# verify its new value:
if { [openCarac_gnucapGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.10.2.7 openCarac_gnucapGetCaseSensitivity

Returns the value of "case sensitivity" attribute for *gnucap* simulator.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac gnucap files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop←FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be set through openCarac_gnucapActivateCaseSensitivity and openCarac_gnucapDeactivateCaseSensitivity.

Returns

Boolean ; 0 if "case sensitivity" attribute of *gnucap* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_gnucapActivateCaseSensitivity

# verify its new value:
if { [openCarac_gnucapGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.10.2.8 openCarac_gnucapGetCheckOptions

Returns the value of "check options" attribute for *gnucap* simulator.

The *gnucap* command is executed by openCarac through the TCL exec command when calling openCarac_running←ExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGet←CustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac←_applicationGetCheckMode. Its value can be set through openCarac_gnucapSetCheckOptions.

Returns

String ; *gnucap* command check options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_gnucapSetCommand      "/usr/bin/gnucap"
openCarac_gnucapSetCheckOptions "-b -n"
openCarac_gnucapSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_gnucapGetCheckOptions]
} else {
    set theOptions [openCarac_gnucapGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_gnucapGetCommand] $theOptions "./mainFile.spi"} fid
```

4.10.2.9 openCarac_gnucapGetCommand

Returns the value of "command" attribute for *gnucap* simulator.

This returns the command to execute *gnucap* simulator. This command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of openCarac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac_gnucapGetCheckOptions and openCarac_gnucapGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be changed through openCarac_gnucapGetCommand.

Returns

String ; *gnucap* command ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_gnucapSetCommand      "/usr/bin/gnucap"
openCarac_gnucapSetCheckOptions "-b -n"
openCarac_gnucapSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_gnucapGetCheckOptions]
} else {
    set theOptions [openCarac_gnucapGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_gnucapGetCommand] $theOptions "./mainFile.spi"} fid
```

4.10.2.10 openCarac_gnucapGetCommentSyntax

Returns the value of "comment syntax" attribute for *gnucap* simulator.

Its value is a non-empty string that is not a list. When creating a temporary folder through openCarac_runningCreate↵TemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be set through openCarac_gnucapSetCommentSyntax.

Returns

String ; comment syntax, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the comment syntax:
openCarac_gnucapSetCommentSyntax "**"

set theComment "[openCarac_gnucapGetCommentSyntax] This is a comment."
puts $theComment
```

4.10.2.11 openCarac_gnucapGetDirectoryChange

Returns the value of "directory change" attribute for *gnucap* simulator.

When executing *gnucap* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *gnucap* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, open↩ Carac changes back the previous location. Its value can be set through openCarac_gnucapActivateDirectoryChange and openCarac_gnucapDeactivateDirectoryChange.

Returns

Boolean ; 0 if "directory change" attribute of *gnucap* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_gnucapActivateDirectoryChange

# verify its new value:
if { [openCarac_gnucapGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.10.2.12 openCarac_gnucapGetIncDirective

Returns the value of "inc directive" attribute for *gnucap* simulator.

Its value is a non-empty string that is not a list ; also, it is different from the gnucap "lib directive" and gnucap "param directive". To define it, case sensitivity depends on the gnucap "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac gnucap "lib directive" and "param directive" can be accessed through openCarac_gnucapGet↩ LibDirective and openCarac_gnucapGetParamDirective. The value of openCarac gnucap "case sensitivity" boolean can be accessed through openCarac_gnucapGetCaseSensitivity. Its value can be set through openCarac_gnucapSetInc↩ Directive.

Returns

String ; inc directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_gnucapSetIncDirective    ".INCLUDE"
openCarac_gnucapSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_gnucapGetIncDirective]
set theDelim     [openCarac_gnucapGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.10.2.13 openCarac_gnucapGetLibDirective

Returns the value of "lib directive" attribute for *gnucap* simulator.

Its value is a non-empty string that is not a list ; also, it is different from the gnucap "inc directive" and gnucap "param directive". To define it, case sensitivity depends on the gnucap "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of openCarac gnucap "inc directive" and "param directive" can be accessed through openCarac_gnucapGetIncDirective and openCarac_gnucap↩GetParamDirective. The value of openCarac gnucap "case sensitivity" boolean can be accessed through openCarac_↩gnucapGetCaseSensitivity. Its value can be set through openCarac_gnucapSetLibDirective.

Returns

String ; lib directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_gnucapSetLibDirective    ".LIB"
openCarac_gnucapSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_gnucapGetLibDirective]
set theDelim     [openCarac_gnucapGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.10.2.14 openCarac_gnucapGetLogExtension

Returns the value of "log extension" attribute for *gnucap* simulator.

This returns the log file extension to print what is returned by the *gnucap* command in. The command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension is a lower case non-empty string, not a list itself, of at least two characters and starting with a dot (.). The value of the *gnucap* command can be accessed through openCarac_gnucapGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be set through openCarac_gnucapSetLogExtension.

Returns

String ; log extension, single word, in lower case, of at least two characters and starting with a dot (.).

Example

```
# change the log extension:
openCarac_gnucapSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_gnucapGetCheckOptions]
} else {
    set theOptions [openCarac_gnucapGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_gnucapGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_gnucapGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.10.2.15 openCarac_gnucapGetMonitorErrorCode

Returns the value of "monitor error code" attribute for *gnucap* simulator.

When executing *gnucap* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *gnucap* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be set through openCarac_gnucapActivateMonitorErrorCode and openCarac_gnucapDeactivateMonitorErrorCode.

Returns

Boolean ; 0 if "monitor error code" attribute of *gnucap* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_gnucapActivateMonitorErrorCode

# verify its new value:
if { [openCarac_gnucapGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.10.2.16 openCarac_gnucapGetParamDirective

Returns the value of "param directive" attribute for *gnucap* simulator.

Its value must be a non-empty string that is not a list ; also, it is different from the gnucap "inc directive" and gnucap "lib directive". To define it, case sensitivity depends on the gnucap "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac gnucap "inc directive" and "lib directive" can be accessed through openCarac_gnucapGetIncDirective and openCarac_gnucapGetLibDirective. The value of openCarac gnucap "case sensitivity" boolean can be accessed through openCarac_gnucapGetCaseSensitivity. The value of "param equality" can be accessed through openCarac_gnucapGetParamEquality. Its value can be set through openCarac_gnucapSetParamDirective.

Returns

String ; param directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_gnucapSetParamDirective ".PARAM"
openCarac_gnucapSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_gnucapGetParamDirective]
set theEqual     [openCarac_gnucapGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.10.2.17 openCarac_gnucapGetParamEquality

Returns the value of "param equality" attribute for *gnucap* simulator.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac gnucap "param directive" can be accessed through openCarac_gnucapGetParamDirective. Its value can be set through openCarac_gnucapSetParamEquality.

Returns

String ; param equality ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_gnucapSetParamDirective ".PARAM"
openCarac_gnucapSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_gnucapGetParamDirective]
set theEqual     [openCarac_gnucapGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.10.2.18 openCarac_gnucapGetRunOptions

Returns the value of "run options" attribute for *gnucap* simulator.

The *gnucap* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_gnucapSetRunOptions.

Returns

String ; *gnucap* command run options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_gnucapSetCommand      "/usr/bin/gnucap"
openCarac_gnucapSetCheckOptions "-b -n"
openCarac_gnucapSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_gnucapGetCheckOptions]
} else {
    set theOptions [openCarac_gnucapGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_gnucapGetCommand] $theOptions "./mainFile.spi"} fid
```

4.10.2.19 openCarac_gnucapGetSaveFilter

Returns the value of "save filter" attribute for *gnucap* simulator.

When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac↩_runningGetSimulatorFilesSavingFolderPath. Save filter is a list of strings in lower case, each of them being a single word starting with a dot (.). The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be set through openCarac_gnucapSetSaveFilter.

Returns

List ; strings in lower case, single words starting with a dot ; integer -1 if an error occurred.

Example

```
set theExtensionsList [openCarac_gnucapGetSaveFilter]

# define which files are not saved by openCarac:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*"] {

    set theExtension [string tolower [file extension $theFile]]

    if { [lsearch $theExtensionsList $theExtension] == -1 } {
        openCarac_warning "This file will not be saved by openCarac: $theFile"
    }

}
```

4.10.2.20 openCarac_gnucapGetStringDelimiter

Returns the value of "string delimiter" attribute for *gnucap* simulator.

Its value is an empty string or a single character. When creating a temporary folder through openCarac_runningCreate↩TemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be set through openCarac_gnucap↩SetStringDelimiter.

Returns

String ; string delimiter, empty string or single character ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_gnucapSetIncDirective    ".INCLUDE"
openCarac_gnucapSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_gnucapGetIncDirective]
set theDelim     [openCarac_gnucapGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.10.2.21 openCarac_gnucapGetToRemoveInCheckMode

Returns the value of "to remove in check mode" attribute for *gnucap* simulator.

This is a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and gnucap comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the gnucap "case sensitivity" attribute (accessible through openCarac_gnucapGetCaseSensitivity). The value of open↩Carac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_gnucapSetToRemoveInCheckMode.

Returns

List ; Strings that are not empty and not lists themselves ; integer -1 if an error occurred.

Example

```
# set the list of directives to remove:
openCarac_gnucapSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_gnucapGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

4.10.2.22 openCarac_gnucapSetCheckOptions *value*

Sets the value of "check options" attribute for *gnucap* simulator.

The *gnucap* command is executed by openCarac through the TCL exec command when calling openCarac_running↩ExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGet↩CustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac↩_applicationGetCheckMode. Its value can be accessed through openCarac_gnucapGetCheckOptions.

Parameters

value	: String ; *gnucap* command check options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_gnucapSetCommand      "/usr/bin/gnucap"
openCarac_gnucapSetCheckOptions "-b -n"
openCarac_gnucapSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_gnucapGetCheckOptions]
} else {
    set theOptions [openCarac_gnucapGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_gnucapGetCommand] $theOptions "./mainFile.spi"} fid
```

4.10.2.23 openCarac_gnucapSetCommand *value*

Sets the value of "command" attribute for *gnucap* simulator.

This sets the command to execute *gnucap* simulator. This command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of openCarac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac_gnucapGetCheckOptions and openCarac_gnucapGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be accessed through openCarac_gnucapGetCommand.

Parameters

value	: String ; *gnucap* command.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_gnucapSetCommand      "/usr/bin/gnucap"
openCarac_gnucapSetCheckOptions "-b -n"
openCarac_gnucapSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_gnucapGetCheckOptions]
} else {
    set theOptions [openCarac_gnucapGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_gnucapGetCommand] $theOptions "./mainFile.spi"} fid
```

4.10.2.24 openCarac_gnucapSetCommentSyntax *value*

Sets the value of "comment syntax" attribute for *gnucap* simulator.

Its value must be a non-empty string that is not a list. When creating a temporary folder through openCarac_running↩ CreateTemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be accessed through openCarac_gnucapGet↩ CommentSyntax.

Parameters

value	: String ; comment syntax, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the comment syntax:
openCarac_gnucapSetCommentSyntax "**"

set theComment "[openCarac_gnucapGetCommentSyntax] This is a comment."
puts $theComment
```

4.10.2.25 openCarac_gnucapSetIncDirective *value*

Sets the value of "inc directive" attribute for *gnucap* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the gnucap "lib directive" and gnucap "param directive". To define it, case sensitivity depends on the gnucap "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac gnucap "lib directive" and "param directive" can be accessed through openCarac_gnucap↩ GetLibDirective and openCarac_gnucapGetParamDirective. The value of openCarac gnucap "case sensitivity" boolean can be accessed through openCarac_gnucapGetCaseSensitivity. Its value can be accessed through openCarac_gnucapGet↩ IncDirective.

Parameters

value	: String ; inc directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_gnucapSetIncDirective    ".INCLUDE"
openCarac_gnucapSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_gnucapGetIncDirective]
set theDelim     [openCarac_gnucapGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.10.2.26 openCarac_gnucapSetLibDirective *value*

Sets the value of "lib directive" attribute for *gnucap* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the gnucap "inc directive" and gnucap "param directive". To define it, case sensitivity depends on the gnucap "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of openCarac gnucap "inc directive" and "param directive" can be accessed through openCarac_gnucapGetIncDirective and openCarac_gnucapGetParamDirective. The value of openCarac gnucap "case sensitivity" boolean can be accessed through openCarac_gnucapGetCaseSensitivity. Its value can be accessed through openCarac_gnucapGetLibDirective.

Parameters

value	: String ; lib directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_gnucapSetLibDirective    ".LIB"
openCarac_gnucapSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_gnucapGetLibDirective]
set theDelim     [openCarac_gnucapGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.10.2.27 openCarac_gnucapSetLogExtension *value*

Sets the value of "log extension" attribute for *gnucap* simulator.

This sets the log file extension to print what is returned by the *gnucap* command in. The command is executed by open↩ Carac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension must be a non-empty string, not a list itself, of at least two characters and starting with a dot (.), openCarac automatically converts it to lower case. If the log file extension does not appear in the "save filter" attribute of openCarac *gnucap* simulator, accessible through openCarac_gnucapGetSaveFilter, it is automatically added. The value of the *gnucap* command can be accessed through openCarac_gnucapGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be accessed through openCarac_gnucapGet↩ LogExtension.

Parameters

value	: String ; log extension, single word, of at least two characters and starting with a dot (.).

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the log extension:
openCarac_gnucapSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_gnucapGetCheckOptions]
} else {
    set theOptions [openCarac_gnucapGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_gnucapGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_gnucapGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.10.2.28 openCarac_gnucapSetParamDirective *value*

Sets the value of "param directive" attribute for *gnucap* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the gnucap "inc directive" and gnucap "lib directive". To define it, case sensitivity depends on the gnucap "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac gnucap "inc directive" and "lib directive" can be accessed through openCarac_gnucapGetIncDirective and openCarac_gnucapGetLibDirective. The value of openCarac gnucap "case sensitivity" boolean can be accessed through openCarac_gnucapGetCaseSensitivity. The value of "param equality" can be accessed through openCarac_gnucapGetParamEquality. Its value can be accessed through openCarac_gnucapGetParamDirective.

Parameters

value	: String ; param directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_gnucapSetParamDirective ".PARAM"
openCarac_gnucapSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_gnucapGetParamDirective]
set theEqual     [openCarac_gnucapGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.10.2.29 openCarac_gnucapSetParamEquality *value*

Sets the value of "param equality" attribute for *gnucap* simulator.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac gnucap "param directive" can be accessed through openCarac_gnucapGetParamDirective. Its value can be accessed through openCarac_gnucapGetParamEquality.

Parameters

value	: String ; param equality.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_gnucapSetParamDirective ".PARAM"
openCarac_gnucapSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_gnucapGetParamDirective]
set theEqual     [openCarac_gnucapGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.10.2.30 openCarac_gnucapSetRunOptions *value*

Sets the value of "run options" attribute for *gnucap* simulator.

The *gnucap* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_gnucapGetRunOptions.

Parameters

value	: String ; *gnucap* command run options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_gnucapSetCommand      "/usr/bin/gnucap"
openCarac_gnucapSetCheckOptions "-b -n"
openCarac_gnucapSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_gnucapGetCheckOptions]
} else {
    set theOptions [openCarac_gnucapGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_gnucapGetCommand] $theOptions "./mainFile.spi"} fid
```

4.10.2.31 openCarac_gnucapSetSaveFilter *value*

Sets the value of "save filter" attribute for *gnucap* simulator.

This must be a list of strings, each of them being a single word starting with a dot (.), openCarac automatically converts them to lower case. When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac_runningGetSimulatorFilesSavingFolderPath. The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be accessed through openCarac↩_gnucapGetSaveFilter.

Parameters

value	: List ; strings, single words starting with a dot.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
set theFilesExtensionsFilter [openCarac_applicationGetFilesExtensionFilter]
set theSaveFilterList        [list]

# use the extensions of the files in the current directory:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*.*"] {

    set theExtension [string tolower [file extension $theFile]]

    # ignore the files that have been copied by openCarac:
    if { [lsearch $theFilesExtensionsFilter $theExtension] == -1 } {
        continue
    }

    if { [lsearch $theSaveFilterList $theExtension] == -1 } {
        lappend theSaveFilterList $theExtension
    }

}

# apply this filter to openCarac:
openCarac_gnucapSetSaveFilter $theSaveFilterList
```

4.10.2.32 openCarac_gnucapSetStringDelimiter *value*

Sets the value of "string delimiter" attribute for *gnucap* simulator.

Its value must be an empty string or a single character. When creating a temporary folder through openCarac_running↩CreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be accessed through openCarac_gnucapGetStringDelimiter.

Parameters

value	: String ; string delimiter, empty string or single character.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_gnucapSetIncDirective    ".INCLUDE"
openCarac_gnucapSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_gnucapGetIncDirective]
set theDelim     [openCarac_gnucapGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.10.2.33 openCarac_gnucapSetToRemoveInCheckMode *value*

Sets the value of "to remove in check mode" attribute for *gnucap* simulator.

This must be a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and gnucap comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the gnucap "case sensitivity" attribute (accessible through openCarac_gnucapGetCaseSensitivity). The value of open↩ Carac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_gnucapGetToRemoveInCheckMode.

Parameters

value	: List ; Strings that are not empty and not lists themselves.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# set the list of directives to remove:
openCarac_gnucapSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_gnucapGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

4.11 Xyce simulator

Definition of functions to interact with openCarac settings for *xyce* simulator.

Functions

- openCarac_xyceGetCommand

  Returns the value of "command" attribute for xyce simulator.
- openCarac_xyceSetCommand value

  Sets the value of "command" attribute for xyce simulator.
- openCarac_xyceGetCheckOptions

  Returns the value of "check options" attribute for xyce simulator.
- openCarac_xyceSetCheckOptions value

  Sets the value of "check options" attribute for xyce simulator.
- openCarac_xyceGetRunOptions

  Returns the value of "run options" attribute for xyce simulator.
- openCarac_xyceSetRunOptions value

  Sets the value of "run options" attribute for xyce simulator.
- openCarac_xyceGetLogExtension

  Returns the value of "log extension" attribute for xyce simulator.
- openCarac_xyceSetLogExtension value

  Sets the value of "log extension" attribute for xyce simulator.
- openCarac_xyceGetSaveFilter

  Returns the value of "save filter" attribute for xyce simulator.
- openCarac_xyceSetSaveFilter value

  Sets the value of "save filter" attribute for xyce simulator.
- openCarac_xyceGetToRemoveInCheckMode

  Returns the value of "to remove in check mode" attribute for xyce simulator.
- openCarac_xyceSetToRemoveInCheckMode value

  Sets the value of "to remove in check mode" attribute for xyce simulator.
- openCarac_xyceGetCommentSyntax

  Returns the value of "comment syntax" attribute for xyce simulator.
- openCarac_xyceSetCommentSyntax value

  Sets the value of "comment syntax" attribute for xyce simulator.
- openCarac_xyceGetIncDirective

  Returns the value of "inc directive" attribute for xyce simulator.
- openCarac_xyceSetIncDirective value

  Sets the value of "inc directive" attribute for xyce simulator.
- openCarac_xyceGetLibDirective

  Returns the value of "lib directive" attribute for xyce simulator.
- openCarac_xyceSetLibDirective value

  Sets the value of "lib directive" attribute for xyce simulator.
- openCarac_xyceGetParamDirective

  Returns the value of "param directive" attribute for xyce simulator.
- openCarac_xyceSetParamDirective value

  Sets the value of "param directive" attribute for xyce simulator.
- openCarac_xyceGetParamEquality

  Returns the value of "param equality" attribute for xyce simulator.

- openCarac_xyceSetParamEquality value

  Sets the value of "param equality" attribute for xyce simulator.
- openCarac_xyceGetStringDelimiter

  Returns the value of "string delimiter" attribute for xyce simulator.
- openCarac_xyceSetStringDelimiter value

  Sets the value of "string delimiter" attribute for xyce simulator.
- openCarac_xyceActivateCaseSensitivity

  Sets openCarac xyce boolean "case sensitivity" to "1".
- openCarac_xyceDeactivateCaseSensitivity

  Sets openCarac xyce boolean "case sensitivity" to "0".
- openCarac_xyceGetCaseSensitivity

  Returns the value of "case sensitivity" attribute for xyce simulator.
- openCarac_xyceActivateDirectoryChange

  Sets openCarac xyce boolean "directory change" to "1".
- openCarac_xyceDeactivateDirectoryChange

  Sets openCarac xyce boolean "directory change" to "0".
- openCarac_xyceGetDirectoryChange

  Returns the value of "directory change" attribute for xyce simulator.
- openCarac_xyceActivateMonitorErrorCode

  Sets openCarac xyce boolean "monitor error code" to "1".
- openCarac_xyceDeactivateMonitorErrorCode

  Sets openCarac xyce boolean "monitor error code" to "0".
- openCarac_xyceGetMonitorErrorCode

  Returns the value of "monitor error code" attribute for xyce simulator.

4.11.1 Detailed Description

Definition of functions to interact with openCarac settings for *xyce* simulator.

openCarac aims to be compatible with various spice simulators. Since different simulators have different syntax and no Spice parser is available in openCarac, a configuration must be done for openCarac to execute it properly. In this module are described every function used to make openCarac fully compatible with *xyce* simulator.

4.11.2 Function Documentation

4.11.2.1 openCarac_xyceActivateCaseSensitivity

Sets openCarac xyce boolean "case sensitivity" to "1".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac xyce files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop←FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_xyceGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_xyceActivateCaseSensitivity

# verify its new value:
if { [openCarac_xyceGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.11.2.2 openCarac_xyceActivateDirectoryChange

Sets openCarac xyce boolean "directory change" to "1".

When executing *xyce* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *xyce* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_xyceGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_xyceActivateDirectoryChange

# verify its new value:
if { [openCarac_xyceGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.11.2.3 openCarac_xyceActivateMonitorErrorCode

Sets openCarac xyce boolean "monitor error code" to "1".

When executing *xyce* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *xyce* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_xyceGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_xyceActivateMonitorErrorCode

# verify its new value:
if { [openCarac_xyceGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.11.2.4 openCarac_xyceDeactivateCaseSensitivity

Sets openCarac xyce boolean "case sensitivity" to "0".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac xyce files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop↵FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_xyceGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_xyceDeactivateCaseSensitivity

# verify its new value:
if { [openCarac_xyceGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.11.2.5 openCarac_xyceDeactivateDirectoryChange

Sets openCarac xyce boolean "directory change" to "0".

When executing *xyce* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *xyce* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_xyceGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_xyceDeactivateDirectoryChange

# verify its new value:
if { [openCarac_xyceGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.11.2.6 openCarac_xyceDeactivateMonitorErrorCode

Sets openCarac xyce boolean "monitor error code" to "0".

When executing *xyce* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *xyce* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_xyceGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_xyceDeactivateMonitorErrorCode

# verify its new value:
if { [openCarac_xyceGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.11.2.7 openCarac_xyceGetCaseSensitivity

Returns the value of "case sensitivity" attribute for *xyce* simulator.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac xyce files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractopFilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be set through openCarac_xyceActivateCaseSensitivity and openCarac_xyceDeactivateCaseSensitivity.

Returns

Boolean ; 0 if "case sensitivity" attribute of *xyce* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_xyceActivateCaseSensitivity

# verify its new value:
if { [openCarac_xyceGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.11.2.8 openCarac_xyceGetCheckOptions

Returns the value of "check options" attribute for *xyce* simulator.

The *xyce* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_xyceSetCheckOptions.

Returns

String ; *xyce* command check options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_xyceSetCommand      "/usr/bin/xyce"
openCarac_xyceSetCheckOptions "-b -n"
openCarac_xyceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_xyceGetCheckOptions]
} else {
    set theOptions [openCarac_xyceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_xyceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.11.2.9 openCarac_xyceGetCommand

Returns the value of "command" attribute for *xyce* simulator.

This returns the command to execute *xyce* simulator. This command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of openCarac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac_xyceGetCheckOptions and openCarac_xyceGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be changed through openCarac_xyceGetCommand.

Returns

String ; *xyce* command ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_xyceSetCommand      "/usr/bin/xyce"
openCarac_xyceSetCheckOptions "-b -n"
openCarac_xyceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_xyceGetCheckOptions]
} else {
    set theOptions [openCarac_xyceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_xyceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.11.2.10 openCarac_xyceGetCommentSyntax

Returns the value of "comment syntax" attribute for *xyce* simulator.

Its value is a non-empty string that is not a list. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be set through openCarac_xyceSetCommentSyntax.

Returns

String ; comment syntax, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the comment syntax:
openCarac_xyceSetCommentSyntax "**"

set theComment "[openCarac_xyceGetCommentSyntax] This is a comment."
puts $theComment
```

4.11.2.11 openCarac_xyceGetDirectoryChange

Returns the value of "directory change" attribute for *xyce* simulator.

When executing *xyce* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *xyce* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be set through openCarac_xyceActivateDirectoryChange and openCarac_xyceDeactivateDirectoryChange.

Returns

Boolean ; 0 if "directory change" attribute of *xyce* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_xyceActivateDirectoryChange

# verify its new value:
if { [openCarac_xyceGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.11.2.12 openCarac_xyceGetIncDirective

Returns the value of "inc directive" attribute for *xyce* simulator.

Its value is a non-empty string that is not a list ; also, it is different from the xyce "lib directive" and xyce "param directive". To define it, case sensitivity depends on the xyce "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac xyce "lib directive" and "param directive" can be accessed through openCarac_xyceGetLibDirective and openCarac_xyceGetParamDirective. The value of openCarac xyce "case sensitivity" boolean can be accessed through openCarac_xyceGetCaseSensitivity. Its value can be set through openCarac_xyceSetIncDirective.

Returns

String ; inc directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_xyceSetIncDirective     ".INCLUDE"
openCarac_xyceSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_xyceGetIncDirective]
set theDelim     [openCarac_xyceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.11.2.13 openCarac_xyceGetLibDirective

Returns the value of "lib directive" attribute for *xyce* simulator.

Its value is a non-empty string that is not a list ; also, it is different from the xyce "inc directive" and xyce "param directive". To define it, case sensitivity depends on the xyce "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of openCarac xyce "inc directive" and "param directive" can be accessed through openCarac_xyceGetIncDirective and openCarac_xyceGetParamDirective. The value of openCarac xyce "case sensitivity" boolean can be accessed through openCarac_xyceGetCaseSensitivity. Its value can be set through openCarac_xyceSetLibDirective.

Returns

String ; lib directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_xyceSetLibDirective     ".LIB"
openCarac_xyceSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_xyceGetLibDirective]
set theDelim     [openCarac_xyceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.11.2.14 openCarac_xyceGetLogExtension

Returns the value of "log extension" attribute for *xyce* simulator.

This returns the log file extension to print what is returned by the *xyce* command in. The command is executed by open↩ Carac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension is a lower case non-empty string, not a list itself, of at least two characters and starting with a dot (.). The value of the *xyce* command can be accessed through openCarac_xyceGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be set through openCarac_xyceSetLog↩ Extension.

Returns

String ; log extension, single word, in lower case, of at least two characters and starting with a dot (.).

Example

```
# change the log extension:
openCarac_xyceSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_xyceGetCheckOptions]
} else {
    set theOptions [openCarac_xyceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_xyceGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_xyceGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.11.2.15 openCarac_xyceGetMonitorErrorCode

Returns the value of "monitor error code" attribute for *xyce* simulator.

When executing *xyce* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *xyce* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be set through openCarac_xyceActivateMonitorErrorCode and openCarac_xyceDeactivateMonitorErrorCode.

Returns

Boolean ; 0 if "monitor error code" attribute of *xyce* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_xyceActivateMonitorErrorCode

# verify its new value:
if { [openCarac_xyceGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.11.2.16 openCarac_xyceGetParamDirective

Returns the value of "param directive" attribute for *xyce* simulator.

Its value must be a non-empty string that is not a list ; also, it is different from the xyce "inc directive" and xyce "lib directive". To define it, case sensitivity depends on the xyce "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac xyce "inc directive" and "lib directive" can be accessed through openCarac_xyceGetIncDirective and openCarac_xyceGetLibDirective. The value of openCarac xyce "case sensitivity" boolean can be accessed through openCarac_xyceGetCaseSensitivity. The value of "param equality" can be accessed through openCarac_xyceGetParamEquality. Its value can be set through openCarac↩_xyceSetParamDirective.

Returns

String ; param directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_xyceSetParamDirective ".PARAM"
openCarac_xyceSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_xyceGetParamDirective]
set theEqual     [openCarac_xyceGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.11.2.17 openCarac_xyceGetParamEquality

Returns the value of "param equality" attribute for *xyce* simulator.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac xyce "param directive" can be accessed through openCarac_xyceGetParamDirective. Its value can be set through openCarac_xyceSetParamEquality.

Returns

String ; param equality ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_xyceSetParamDirective ".PARAM"
openCarac_xyceSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_xyceGetParamDirective]
set theEqual     [openCarac_xyceGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.11.2.18 openCarac_xyceGetRunOptions

Returns the value of "run options" attribute for *xyce* simulator.

The *xyce* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_xyceSetRunOptions.

Returns

String ; *xyce* command run options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_xyceSetCommand      "/usr/bin/xyce"
openCarac_xyceSetCheckOptions "-b -n"
openCarac_xyceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_xyceGetCheckOptions]
} else {
    set theOptions [openCarac_xyceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_xyceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.11.2.19 openCarac_xyceGetSaveFilter

Returns the value of "save filter" attribute for *xyce* simulator.

When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac_runningGetSimulatorFilesSavingFolderPath. Save filter is a list of strings in lower case, each of them being a single word starting with a dot (.). The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be set through openCarac_xyceSetSaveFilter.

Returns

List ; strings in lower case, single words starting with a dot ; integer -1 if an error occurred.

Example

```
set theExtensionsList [openCarac_xyceGetSaveFilter]

# define which files are not saved by openCarac:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*"] {

    set theExtension [string tolower [file extension $theFile]]

    if { [lsearch $theExtensionsList $theExtension] == -1 } {
        openCarac_warning "This file will not be saved by openCarac: $theFile"
    }

}
```

4.11.2.20 openCarac_xyceGetStringDelimiter

Returns the value of "string delimiter" attribute for *xyce* simulator.

Its value is an empty string or a single character. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be set through openCarac_xyceSetStringDelimiter.

Returns

String ; string delimiter, empty string or single character ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_xyceSetIncDirective     ".INCLUDE"
openCarac_xyceSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_xyceGetIncDirective]
set theDelim     [openCarac_xyceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.11.2.21 openCarac_xyceGetToRemoveInCheckMode

Returns the value of "to remove in check mode" attribute for *xyce* simulator.

This is a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and xyce comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the xyce "case sensitivity" attribute (accessible through openCarac_xyceGetCaseSensitivity). The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_xyceSetToRemoveInCheckMode.

Returns

List ; Strings that are not empty and not lists themselves ; integer -1 if an error occurred.

Example

```
# set the list of directives to remove:
openCarac_xyceSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_xyceGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

4.11.2.22 openCarac_xyceSetCheckOptions *value*

Sets the value of "check options" attribute for *xyce* simulator.

The *xyce* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecute↩ Simulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustom↩ ExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_↩ applicationGetCheckMode. Its value can be accessed through openCarac_xyceGetCheckOptions.

Parameters

value	: String ; *xyce* command check options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_xyceSetCommand      "/usr/bin/xyce"
openCarac_xyceSetCheckOptions "-b -n"
openCarac_xyceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_xyceGetCheckOptions]
} else {
    set theOptions [openCarac_xyceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_xyceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.11.2.23 openCarac_xyceSetCommand *value*

Sets the value of "command" attribute for *xyce* simulator.

This sets the command to execute *xyce* simulator. This command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of openCarac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac_xyceGetCheckOptions and openCarac_xyceGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be accessed through openCarac_xyceGetCommand.

Parameters

value	: String ; *xyce* command.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_xyceSetCommand      "/usr/bin/xyce"
openCarac_xyceSetCheckOptions "-b -n"
openCarac_xyceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_xyceGetCheckOptions]
} else {
    set theOptions [openCarac_xyceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_xyceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.11.2.24 openCarac_xyceSetCommentSyntax *value*

Sets the value of "comment syntax" attribute for *xyce* simulator.

Its value must be a non-empty string that is not a list. When creating a temporary folder through openCarac_running↩ CreateTemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be accessed through openCarac_xyceGetComment↩ Syntax.

Parameters

value	: String ; comment syntax, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the comment syntax:
openCarac_xyceSetCommentSyntax "**"

set theComment "[openCarac_xyceGetCommentSyntax] This is a comment."
puts $theComment
```

4.11.2.25 openCarac_xyceSetIncDirective *value*

Sets the value of "inc directive" attribute for *xyce* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the xyce "lib directive" and xyce "param directive". To define it, case sensitivity depends on the xyce "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac xyce "lib directive" and "param directive" can be accessed through openCarac_xyceGetLibDirective and openCarac_xyceGetParamDirective. The value of openCarac xyce "case sensitivity" boolean can be accessed through openCarac_xyceGetCaseSensitivity. Its value can be accessed through openCarac_xyceGetIncDirective.

Parameters

value	: String ; inc directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_xyceSetIncDirective     ".INCLUDE"
openCarac_xyceSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_xyceGetIncDirective]
set theDelim     [openCarac_xyceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.11.2.26 openCarac_xyceSetLibDirective *value*

Sets the value of "lib directive" attribute for *xyce* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the xyce "inc directive" and xyce "param directive". To define it, case sensitivity depends on the xyce "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of openCarac xyce "inc directive" and "param directive" can be accessed through openCarac_xyceGetIncDirective and openCarac_xyceGetParamDirective. The value of openCarac xyce "case sensitivity" boolean can be accessed through openCarac_xyceGetCaseSensitivity. Its value can be accessed through openCarac_xyceGetLibDirective.

Parameters

value	: String ; lib directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_xyceSetLibDirective    ".LIB"
openCarac_xyceSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_xyceGetLibDirective]
set theDelim     [openCarac_xyceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.11.2.27 openCarac_xyceSetLogExtension *value*

Sets the value of "log extension" attribute for *xyce* simulator.

This sets the log file extension to print what is returned by the *xyce* command in. The command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension must be a non-empty string, not a list itself, of at least two characters and starting with a dot (.), openCarac automatically converts it to lower case. If the log file extension does not appear in the "save filter" attribute of openCarac *xyce* simulator, accessible through openCarac_xyceGetSaveFilter, it is automatically added. The value of the *xyce* command can be accessed through openCarac_xyceGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be accessed through openCarac_xyceGetLogExtension.

Parameters

value	: String ; log extension, single word, of at least two characters and starting with a dot (.).

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the log extension:
openCarac_xyceSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_xyceGetCheckOptions]
} else {
    set theOptions [openCarac_xyceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_xyceGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_xyceGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.11.2.28 openCarac_xyceSetParamDirective *value*

Sets the value of "param directive" attribute for *xyce* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the xyce "inc directive" and xyce "lib directive". To define it, case sensitivity depends on the xyce "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac xyce "inc directive" and "lib directive" can be accessed through openCarac_xyceGetIncDirective and openCarac_xyceGetLibDirective. The value of openCarac xyce "case sensitivity" boolean can be accessed through openCarac_xyceGetCaseSensitivity. The value of "param equality" can be accessed through openCarac_xyceGetParamEquality. Its value can be accessed through openCarac_xyceGetParamDirective.

Parameters

value	: String ; param directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_xyceSetParamDirective ".PARAM"
openCarac_xyceSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_xyceGetParamDirective]
set theEqual     [openCarac_xyceGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.11.2.29 openCarac_xyceSetParamEquality *value*

Sets the value of "param equality" attribute for *xyce* simulator.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac xyce "param directive" can be accessed through openCarac_xyceGetParamDirective. Its value can be accessed through openCarac_xyceGetParamEquality.

Parameters

value	: String ; param equality.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_xyceSetParamDirective ".PARAM"
openCarac_xyceSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_xyceGetParamDirective]
set theEqual     [openCarac_xyceGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.11.2.30 openCarac_xyceSetRunOptions *value*

Sets the value of "run options" attribute for *xyce* simulator.

The *xyce* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_xyceGetRunOptions.

Parameters

value	: String ; *xyce* command run options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_xyceSetCommand      "/usr/bin/xyce"
openCarac_xyceSetCheckOptions "-b -n"
openCarac_xyceSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_xyceGetCheckOptions]
} else {
    set theOptions [openCarac_xyceGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_xyceGetCommand] $theOptions "./mainFile.spi"} fid
```

4.11.2.31 openCarac_xyceSetSaveFilter *value*

Sets the value of "save filter" attribute for *xyce* simulator.

This must be a list of strings, each of them being a single word starting with a dot (.), openCarac automatically converts them to lower case. When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac_runningGetSimulatorFilesSavingFolderPath. The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be accessed through openCarac_xyceGetSaveFilter.

Parameters

value	: List ; strings, single words starting with a dot.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
set theFilesExtensionsFilter [openCarac_applicationGetFilesExtensionFilter]
set theSaveFilterList        [list]

# use the extensions of the files in the current directory:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*.*"] {

    set theExtension [string tolower [file extension $theFile]]

    # ignore the files that have been copied by openCarac:
    if { [lsearch $theFilesExtensionsFilter $theExtension] == -1 } {
        continue
    }

    if { [lsearch $theSaveFilterList $theExtension] == -1 } {
        lappend theSaveFilterList $theExtension
    }

}

# apply this filter to openCarac:
openCarac_xyceSetSaveFilter $theSaveFilterList
```

4.11.2.32 openCarac_xyceSetStringDelimiter *value*

Sets the value of "string delimiter" attribute for *xyce* simulator.

Its value must be an empty string or a single character. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be accessed through openCarac_xyceGetStringDelimiter.

Parameters

value	: String ; string delimiter, empty string or single character.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_xyceSetIncDirective    ".INCLUDE"
openCarac_xyceSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_xyceGetIncDirective]
set theDelim     [openCarac_xyceGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.11.2.33 openCarac_xyceSetToRemoveInCheckMode *value*

Sets the value of "to remove in check mode" attribute for *xyce* simulator.

This must be a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and xyce comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the xyce "case sensitivity" attribute (accessible through openCarac_xyceGetCaseSensitivity). The value of open↵ Carac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_xyceGetToRemoveInCheckMode.

Parameters

value	: List ; Strings that are not empty and not lists themselves.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# set the list of directives to remove:
openCarac_xyceSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_xyceGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

4.12 Smash simulator

Definition of functions to interact with openCarac settings for *smash* simulator.

Functions

- openCarac_smashGetCommand

  Returns the value of "command" attribute for smash simulator.
- openCarac_smashSetCommand value

  Sets the value of "command" attribute for smash simulator.
- openCarac_smashGetCheckOptions

  Returns the value of "check options" attribute for smash simulator.
- openCarac_smashSetCheckOptions value

  Sets the value of "check options" attribute for smash simulator.
- openCarac_smashGetRunOptions

  Returns the value of "run options" attribute for smash simulator.
- openCarac_smashSetRunOptions value

  Sets the value of "run options" attribute for smash simulator.
- openCarac_smashGetLogExtension

  Returns the value of "log extension" attribute for smash simulator.
- openCarac_smashSetLogExtension value

  Sets the value of "log extension" attribute for smash simulator.
- openCarac_smashGetSaveFilter

  Returns the value of "save filter" attribute for smash simulator.
- openCarac_smashSetSaveFilter value

  Sets the value of "save filter" attribute for smash simulator.
- openCarac_smashGetToRemoveInCheckMode

  Returns the value of "to remove in check mode" attribute for smash simulator.
- openCarac_smashSetToRemoveInCheckMode value

  Sets the value of "to remove in check mode" attribute for smash simulator.
- openCarac_smashGetCommentSyntax

  Returns the value of "comment syntax" attribute for smash simulator.
- openCarac_smashSetCommentSyntax value

  Sets the value of "comment syntax" attribute for smash simulator.
- openCarac_smashGetIncDirective

  Returns the value of "inc directive" attribute for smash simulator.
- openCarac_smashSetIncDirective value

  Sets the value of "inc directive" attribute for smash simulator.
- openCarac_smashGetLibDirective

  Returns the value of "lib directive" attribute for smash simulator.
- openCarac_smashSetLibDirective value

  Sets the value of "lib directive" attribute for smash simulator.
- openCarac_smashGetParamDirective

  Returns the value of "param directive" attribute for smash simulator.
- openCarac_smashSetParamDirective value

  Sets the value of "param directive" attribute for smash simulator.
- openCarac_smashGetParamEquality

  Returns the value of "param equality" attribute for smash simulator.

- openCarac_smashSetParamEquality value

  Sets the value of "param equality" attribute for smash simulator.
- openCarac_smashGetStringDelimiter

  Returns the value of "string delimiter" attribute for smash simulator.
- openCarac_smashSetStringDelimiter value

  Sets the value of "string delimiter" attribute for smash simulator.
- openCarac_smashActivateCaseSensitivity

  Sets openCarac smash boolean "case sensitivity" to "1".
- openCarac_smashDeactivateCaseSensitivity

  Sets openCarac smash boolean "case sensitivity" to "0".
- openCarac_smashGetCaseSensitivity

  Returns the value of "case sensitivity" attribute for smash simulator.
- openCarac_smashActivateDirectoryChange

  Sets openCarac smash boolean "directory change" to "1".
- openCarac_smashDeactivateDirectoryChange

  Sets openCarac smash boolean "directory change" to "0".
- openCarac_smashGetDirectoryChange

  Returns the value of "directory change" attribute for smash simulator.
- openCarac_smashActivateMonitorErrorCode

  Sets openCarac smash boolean "monitor error code" to "1".
- openCarac_smashDeactivateMonitorErrorCode

  Sets openCarac smash boolean "monitor error code" to "0".
- openCarac_smashGetMonitorErrorCode

  Returns the value of "monitor error code" attribute for smash simulator.

4.12.1 Detailed Description

Definition of functions to interact with openCarac settings for *smash* simulator.

openCarac aims to be compatible with various spice simulators. Since different simulators have different syntax and no Spice parser is available in openCarac, a configuration must be done for openCarac to execute it properly. In this module are described every function used to make openCarac fully compatible with *smash* simulator.

4.12.2 Function Documentation

4.12.2.1 openCarac_smashActivateCaseSensitivity

Sets openCarac smash boolean "case sensitivity" to "1".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac smash files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop↩ FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_smashGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_smashActivateCaseSensitivity

# verify its new value:
if { [openCarac_smashGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.12.2.2 openCarac_smashActivateDirectoryChange

Sets openCarac smash boolean "directory change" to "1".

When executing *smash* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *smash* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_smashGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_smashActivateDirectoryChange

# verify its new value:
if { [openCarac_smashGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.12.2.3 openCarac_smashActivateMonitorErrorCode

Sets openCarac smash boolean "monitor error code" to "1".

When executing *smash* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *smash* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_smashGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_smashActivateMonitorErrorCode

# verify its new value:
if { [openCarac_smashGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.12.2.4 openCarac_smashDeactivateCaseSensitivity

Sets openCarac smash boolean "case sensitivity" to "0".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac smash files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop↩ FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_smashGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_smashDeactivateCaseSensitivity

# verify its new value:
if { [openCarac_smashGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.12.2.5 openCarac_smashDeactivateDirectoryChange

Sets openCarac smash boolean "directory change" to "0".

When executing *smash* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *smash* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_smashGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_smashDeactivateDirectoryChange

# verify its new value:
if { [openCarac_smashGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.12.2.6 openCarac_smashDeactivateMonitorErrorCode

Sets openCarac smash boolean "monitor error code" to "0".

When executing *smash* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *smash* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_smashGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_smashDeactivateMonitorErrorCode

# verify its new value:
if { [openCarac_smashGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.12.2.7 openCarac_smashGetCaseSensitivity

Returns the value of "case sensitivity" attribute for *smash* simulator.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac smash files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractop↩FilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be set through openCarac_smashActivateCaseSensitivity and openCarac_smashDeactivateCaseSensitivity.

Returns

Boolean ; 0 if "case sensitivity" attribute of *smash* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_smashActivateCaseSensitivity

# verify its new value:
if { [openCarac_smashGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.12.2.8 openCarac_smashGetCheckOptions

Returns the value of "check options" attribute for *smash* simulator.

The *smash* command is executed by openCarac through the TCL exec command when calling openCarac_running↩ExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGet↩CustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac↩_applicationGetCheckMode. Its value can be set through openCarac_smashSetCheckOptions.

Returns

String ; *smash* command check options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_smashSetCommand      "/usr/bin/smash"
openCarac_smashSetCheckOptions "-b -n"
openCarac_smashSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_smashGetCheckOptions]
} else {
    set theOptions [openCarac_smashGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_smashGetCommand] $theOptions "./mainFile.spi"} fid
```

4.12.2.9 openCarac_smashGetCommand

Returns the value of "command" attribute for *smash* simulator.

This returns the command to execute *smash* simulator. This command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of openCarac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac_smashGetCheckOptions and openCarac_smashGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be changed through openCarac_smashGetCommand.

Returns

String ; *smash* command ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_smashSetCommand      "/usr/bin/smash"
openCarac_smashSetCheckOptions "-b -n"
openCarac_smashSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_smashGetCheckOptions]
} else {
    set theOptions [openCarac_smashGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_smashGetCommand] $theOptions "./mainFile.spi"} fid
```

4.12.2.10 openCarac_smashGetCommentSyntax

Returns the value of "comment syntax" attribute for *smash* simulator.

Its value is a non-empty string that is not a list. When creating a temporary folder through openCarac_runningCreate←TemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be set through openCarac_smashSetCommentSyntax.

Returns

String ; comment syntax, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the comment syntax:
openCarac_smashSetCommentSyntax "**"

set theComment "[openCarac_smashGetCommentSyntax] This is a comment."
puts $theComment
```

4.12.2.11 openCarac_smashGetDirectoryChange

Returns the value of "directory change" attribute for *smash* simulator.

When executing *smash* command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the *smash* command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be set through openCarac_smashActivateDirectoryChange and openCarac_smashDeactivateDirectoryChange.

Returns

Boolean ; 0 if "directory change" attribute of *smash* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_smashActivateDirectoryChange

# verify its new value:
if { [openCarac_smashGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.12.2.12 openCarac_smashGetIncDirective

Returns the value of "inc directive" attribute for *smash* simulator.

Its value is a non-empty string that is not a list ; also, it is different from the smash "lib directive" and smash "param directive". To define it, case sensitivity depends on the smash "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac smash "lib directive" and "param directive" can be accessed through openCarac_smashGetLibDirective and openCarac_smashGetParamDirective. The value of openCarac smash "case sensitivity" boolean can be accessed through openCarac_smashGetCaseSensitivity. Its value can be set through openCarac_smashSetIncDirective.

Returns

String ; inc directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_smashSetIncDirective    ".INCLUDE"
openCarac_smashSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_smashGetIncDirective]
set theDelim     [openCarac_smashGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.12.2.13 openCarac_smashGetLibDirective

Returns the value of "lib directive" attribute for *smash* simulator.

Its value is a non-empty string that is not a list ; also, it is different from the smash "inc directive" and smash "param directive". To define it, case sensitivity depends on the smash "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of openCarac smash "inc directive" and "param directive" can be accessed through openCarac_smashGetIncDirective and openCarac_smashGet↩ParamDirective. The value of openCarac smash "case sensitivity" boolean can be accessed through openCarac_smash↩GetCaseSensitivity. Its value can be set through openCarac_smashSetLibDirective.

Returns

String ; lib directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_smashSetLibDirective    ".LIB"
openCarac_smashSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_smashGetLibDirective]
set theDelim     [openCarac_smashGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.12.2.14 openCarac_smashGetLogExtension

Returns the value of "log extension" attribute for *smash* simulator.

This returns the log file extension to print what is returned by the *smash* command in. The command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension is a lower case non-empty string, not a list itself, of at least two characters and starting with a dot (.). The value of the *smash* command can be accessed through openCarac_smashGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be set through openCarac_smashSetLogExtension.

Returns

String ; log extension, single word, in lower case, of at least two characters and starting with a dot (.).

Example

```
# change the log extension:
openCarac_smashSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_smashGetCheckOptions]
} else {
    set theOptions [openCarac_smashGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_smashGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_smashGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.12.2.15 openCarac_smashGetMonitorErrorCode

Returns the value of "monitor error code" attribute for *smash* simulator.

When executing *smash* command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the *smash* command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be set through openCarac_smashActivateMonitorErrorCode and openCarac_smashDeactivateMonitorErrorCode.

Returns

Boolean ; 0 if "monitor error code" attribute of *smash* simulator is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_smashActivateMonitorErrorCode

# verify its new value:
if { [openCarac_smashGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.12.2.16 openCarac_smashGetParamDirective

Returns the value of "param directive" attribute for *smash* simulator.

Its value must be a non-empty string that is not a list ; also, it is different from the smash "inc directive" and smash "lib directive". To define it, case sensitivity depends on the smash "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac smash "inc directive" and "lib directive" can be accessed through openCarac_smashGetIncDirective and openCarac_smashGetLibDirective. The value of openCarac smash "case sensitivity" boolean can be accessed through openCarac_smashGetCaseSensitivity. The value of "param equality" can be accessed through openCarac_smashGetParamEquality. Its value can be set through openCarac_smashSetParamDirective.

Returns

String ; param directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_smashSetParamDirective ".PARAM"
openCarac_smashSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_smashGetParamDirective]
set theEqual     [openCarac_smashGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.12.2.17 openCarac_smashGetParamEquality

Returns the value of "param equality" attribute for *smash* simulator.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac smash "param directive" can be accessed through openCarac_smashGetParamDirective. Its value can be set through openCarac_smashSetParamEquality.

Returns

String ; param equality ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_smashSetParamDirective ".PARAM"
openCarac_smashSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_smashGetParamDirective]
set theEqual     [openCarac_smashGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.12.2.18 openCarac_smashGetRunOptions

Returns the value of "run options" attribute for *smash* simulator.

The *smash* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_smashSetRunOptions.

Returns

String ; *smash* command run options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_smashSetCommand      "/usr/bin/smash"
openCarac_smashSetCheckOptions "-b -n"
openCarac_smashSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_smashGetCheckOptions]
} else {
    set theOptions [openCarac_smashGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_smashGetCommand] $theOptions "./mainFile.spi"} fid
```

4.12.2.19 openCarac_smashGetSaveFilter

Returns the value of "save filter" attribute for *smash* simulator.

When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac_runningGetSimulatorFilesSavingFolderPath. Save filter is a list of strings in lower case, each of them being a single word starting with a dot (.). The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be set through openCarac_smashSetSaveFilter.

Returns

List ; strings in lower case, single words starting with a dot ; integer -1 if an error occurred.

Example

```
set theExtensionsList [openCarac_smashGetSaveFilter]

# define which files are not saved by openCarac:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*"] {

    set theExtension [string tolower [file extension $theFile]]

    if { [lsearch $theExtensionsList $theExtension] == -1 } {
        openCarac_warning "This file will not be saved by openCarac: $theFile"
    }

}
```

4.12.2.20 openCarac_smashGetStringDelimiter

Returns the value of "string delimiter" attribute for *smash* simulator.

Its value is an empty string or a single character. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be set through openCarac_smashSetStringDelimiter.

Returns

String ; string delimiter, empty string or single character ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_smashSetIncDirective     ".INCLUDE"
openCarac_smashSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_smashGetIncDirective]
set theDelim     [openCarac_smashGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.12.2.21 openCarac_smashGetToRemoveInCheckMode

Returns the value of "to remove in check mode" attribute for *smash* simulator.

This is a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and smash comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the smash "case sensitivity" attribute (accessible through openCarac_smashGetCaseSensitivity). The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_smashSetToRemoveInCheckMode.

Returns

List ; Strings that are not empty and not lists themselves ; integer -1 if an error occurred.

Example

```
# set the list of directives to remove:
openCarac_smashSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_smashGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

4.12.2.22 openCarac_smashSetCheckOptions *value*

Sets the value of "check options" attribute for *smash* simulator.

The *smash* command is executed by openCarac through the TCL exec command when calling openCarac_running↩ExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGet↩CustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac↩_applicationGetCheckMode. Its value can be accessed through openCarac_smashGetCheckOptions.

Parameters

value	: String ; *smash* command check options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_smashSetCommand      "/usr/bin/smash"
openCarac_smashSetCheckOptions "-b -n"
openCarac_smashSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_smashGetCheckOptions]
} else {
    set theOptions [openCarac_smashGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_smashGetCommand] $theOptions "./mainFile.spi"} fid
```

4.12.2.23 openCarac_smashSetCommand *value*

Sets the value of "command" attribute for *smash* simulator.

This sets the command to execute *smash* simulator. This command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of openCarac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac_smashGetCheckOptions and openCarac_smashGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be accessed through openCarac_smashGetCommand.

Parameters

value	: String ; *smash* command.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_smashSetCommand      "/usr/bin/smash"
openCarac_smashSetCheckOptions "-b -n"
openCarac_smashSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_smashGetCheckOptions]
} else {
    set theOptions [openCarac_smashGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_smashGetCommand] $theOptions "./mainFile.spi"} fid
```

4.12.2.24 openCarac_smashSetCommentSyntax *value*

Sets the value of "comment syntax" attribute for *smash* simulator.

Its value must be a non-empty string that is not a list. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be accessed through openCarac_smashGetCommentSyntax.

Parameters

value	: String ; comment syntax, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the comment syntax:
openCarac_smashSetCommentSyntax "**"

set theComment "[openCarac_smashGetCommentSyntax] This is a comment."
puts $theComment
```

4.12.2.25 openCarac_smashSetIncDirective *value*

Sets the value of "inc directive" attribute for *smash* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the smash "lib directive" and smash "param directive". To define it, case sensitivity depends on the smash "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac smash "lib directive" and "param directive" can be accessed through openCarac_smashGetLibDirective and openCarac_smashGetParamDirective. The value of openCarac smash "case sensitivity" boolean can be accessed through openCarac_smashGetCaseSensitivity. Its value can be accessed through openCarac_smashGetIncDirective.

Parameters

value	: String ; inc directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_smashSetIncDirective     ".INCLUDE"
openCarac_smashSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_smashGetIncDirective]
set theDelim     [openCarac_smashGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.12.2.26 openCarac_smashSetLibDirective *value*

Sets the value of "lib directive" attribute for *smash* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the smash "inc directive" and smash "param directive". To define it, case sensitivity depends on the smash "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of openCarac smash "inc directive" and "param directive" can be accessed through openCarac_smashGetIncDirective and openCarac_smashGetParamDirective. The value of openCarac smash "case sensitivity" boolean can be accessed through openCarac_smashGetCaseSensitivity. Its value can be accessed through openCarac_smashGetLibDirective.

Parameters

value	: String ; lib directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_smashSetLibDirective     ".LIB"
openCarac_smashSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_smashGetLibDirective]
set theDelim     [openCarac_smashGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.12.2.27 openCarac_smashSetLogExtension *value*

Sets the value of "log extension" attribute for *smash* simulator.

This sets the log file extension to print what is returned by the *smash* command in. The command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension must be a non-empty string, not a list itself, of at least two characters and starting with a dot (.), openCarac automatically converts it to lower case. If the log file extension does not appear in the "save filter" attribute of openCarac *smash* simulator, accessible through openCarac_smashGetSaveFilter, it is automatically added. The value of the *smash* command can be accessed through openCarac_smashGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be accessed through openCarac_smashGetLogExtension.

Parameters

value	: String ; log extension, single word, of at least two characters and starting with a dot (.).

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the log extension:
openCarac_smashSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_smashGetCheckOptions]
} else {
    set theOptions [openCarac_smashGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_smashGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_smashGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.12.2.28 openCarac_smashSetParamDirective *value*

Sets the value of "param directive" attribute for *smash* simulator.

Its value must be a non-empty string that is not a list ; also, it must be different from the smash "inc directive" and smash "lib directive". To define it, case sensitivity depends on the smash "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac smash "inc directive" and "lib directive" can be accessed through openCarac_smashGetIncDirective and openCarac_smashGetLibDirective. The value of openCarac smash "case sensitivity" boolean can be accessed through openCarac_smashGetCaseSensitivity. The value of "param equality" can be accessed through openCarac_smashGetParamEquality. Its value can be accessed through openCarac_smashGetParamDirective.

Parameters

value	: String ; param directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_smashSetParamDirective ".PARAM"
openCarac_smashSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_smashGetParamDirective]
set theEqual     [openCarac_smashGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.12.2.29 openCarac_smashSetParamEquality *value*

Sets the value of "param equality" attribute for *smash* simulator.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac smash "param directive" can be accessed through openCarac_smashGetParamDirective. Its value can be accessed through openCarac_smashGetParamEquality.

Parameters

value	: String ; param equality.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_smashSetParamDirective ".PARAM"
openCarac_smashSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_smashGetParamDirective]
set theEqual     [openCarac_smashGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.12.2.30 openCarac_smashSetRunOptions *value*

Sets the value of "run options" attribute for *smash* simulator.

The *smash* command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_smashGetRunOptions.

Parameters

value	: String ; *smash* command run options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_smashSetCommand      "/usr/bin/smash"
openCarac_smashSetCheckOptions "-b -n"
openCarac_smashSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_smashGetCheckOptions]
} else {
    set theOptions [openCarac_smashGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_smashGetCommand] $theOptions "./mainFile.spi"} fid
```

4.12.2.31 openCarac_smashSetSaveFilter *value*

Sets the value of "save filter" attribute for *smash* simulator.

This must be a list of strings, each of them being a single word starting with a dot (.), openCarac automatically converts them to lower case. When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac_runningGetSimulatorFilesSavingFolderPath. The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be accessed through openCarac_smashGetSaveFilter.

Parameters

value	: List ; strings, single words starting with a dot.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
set theFilesExtensionsFilter [openCarac_applicationGetFilesExtensionFilter]
set theSaveFilterList        [list]

# use the extensions of the files in the current directory:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*.*"] {

    set theExtension [string tolower [file extension $theFile]]

    # ignore the files that have been copied by openCarac:
    if { [lsearch $theFilesExtensionsFilter $theExtension] == -1 } {
        continue
    }

    if { [lsearch $theSaveFilterList $theExtension] == -1 } {
        lappend theSaveFilterList $theExtension
    }

}

# apply this filter to openCarac:
openCarac_smashSetSaveFilter $theSaveFilterList
```

4.12.2.32 openCarac_smashSetStringDelimiter *value*

Sets the value of "string delimiter" attribute for *smash* simulator.

Its value must be an empty string or a single character. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be accessed through openCarac_smashGetStringDelimiter.

Parameters

value	: String ; string delimiter, empty string or single character.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_smashSetIncDirective     ".INCLUDE"
openCarac_smashSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_smashGetIncDirective]
set theDelim     [openCarac_smashGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.12.2.33 openCarac_smashSetToRemoveInCheckMode *value*

Sets the value of "to remove in check mode" attribute for *smash* simulator.

This must be a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and smash comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the smash "case sensitivity" attribute (accessible through openCarac_smashGetCaseSensitivity). The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_smashGetToRemoveInCheckMode.

Parameters

value	: List ; Strings that are not empty and not lists themselves.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# set the list of directives to remove:
openCarac_smashSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_smashGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

4.13 Command class

Definition of functions to interact with openCarac settings for your custom command.

Functions

- openCarac_commandGetCommand

  Returns the value of "command" attribute for your custom command.
- openCarac_commandSetCommand value

  Sets the value of "command" attribute for your custom command.
- openCarac_commandGetCheckOptions

  Returns the value of "check options" attribute for your custom command.
- openCarac_commandSetCheckOptions value

  Sets the value of "check options" attribute for your custom command.
- openCarac_commandGetRunOptions

  Returns the value of "run options" attribute for your custom command.
- openCarac_commandSetRunOptions value

  Sets the value of "run options" attribute for your custom command.
- openCarac_commandGetLogExtension

  Returns the value of "log extension" attribute for your custom command.
- openCarac_commandSetLogExtension value

  Sets the value of "log extension" attribute for your custom command.
- openCarac_commandGetSaveFilter

  Returns the value of "save filter" attribute for your custom command.
- openCarac_commandSetSaveFilter value

  Sets the value of "save filter" attribute for your custom command.
- openCarac_commandGetToRemoveInCheckMode

  Returns the value of "to remove in check mode" attribute for your custom command.
- openCarac_commandSetToRemoveInCheckMode value

  Sets the value of "to remove in check mode" attribute for your custom command.
- openCarac_commandGetCommentSyntax

  Returns the value of "comment syntax" attribute for your custom command.
- openCarac_commandSetCommentSyntax value

  Sets the value of "comment syntax" attribute for your custom command.
- openCarac_commandGetIncDirective

  Returns the value of "inc directive" attribute for your custom command.
- openCarac_commandSetIncDirective value

  Sets the value of "inc directive" attribute for your custom command.
- openCarac_commandGetLibDirective

  Returns the value of "lib directive" attribute for your custom command.
- openCarac_commandSetLibDirective value

  Sets the value of "lib directive" attribute for your custom command.
- openCarac_commandGetParamDirective

  Returns the value of "param directive" attribute for your custom command.
- openCarac_commandSetParamDirective value

  Sets the value of "param directive" attribute for your custom command.
- openCarac_commandGetParamEquality

  Returns the value of "param equality" attribute for your custom command.

- openCarac_commandSetParamEquality value

  Sets the value of "param equality" attribute for your custom command.
- openCarac_commandGetStringDelimiter

  Returns the value of "string delimiter" attribute for your custom command.
- openCarac_commandSetStringDelimiter value

  Sets the value of "string delimiter" attribute for your custom command.
- openCarac_commandActivateCaseSensitivity

  Sets openCarac command boolean "case sensitivity" to "1".
- openCarac_commandDeactivateCaseSensitivity

  Sets openCarac command boolean "case sensitivity" to "0".
- openCarac_commandGetCaseSensitivity

  Returns the value of "case sensitivity" attribute for your custom command.
- openCarac_commandActivateDirectoryChange

  Sets openCarac command boolean "directory change" to "1".
- openCarac_commandDeactivateDirectoryChange

  Sets openCarac command boolean "directory change" to "0".
- openCarac_commandGetDirectoryChange

  Returns the value of "directory change" attribute for your custom command.
- openCarac_commandActivateMonitorErrorCode

  Sets openCarac command boolean "monitor error code" to "1".
- openCarac_commandDeactivateMonitorErrorCode

  Sets openCarac command boolean "monitor error code" to "0".
- openCarac_commandGetMonitorErrorCode

  Returns the value of "monitor error code" attribute for your custom command.

4.13.1 Detailed Description

Definition of functions to interact with openCarac settings for your custom command.

openCarac aims to be compatible with various spice simulators. Since different simulators have different syntax and no Spice parser is available in openCarac, a configuration must be done for openCarac to execute it properly. In this module are described every function used to make openCarac fully compatible with your custom command.

4.13.2 Function Documentation

4.13.2.1 openCarac_commandActivateCaseSensitivity

Sets openCarac command boolean "case sensitivity" to "1".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac command files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_carac↩GetExtractopFilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_commandGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_commandActivateCaseSensitivity

# verify its new value:
if { [openCarac_commandGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.13.2.2 openCarac_commandActivateDirectoryChange

Sets openCarac command boolean "directory change" to "1".

When executing your custom command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the your custom command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_commandGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_commandActivateDirectoryChange

# verify its new value:
if { [openCarac_commandGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.13.2.3 openCarac_commandActivateMonitorErrorCode

Sets openCarac command boolean "monitor error code" to "1".

When executing your custom command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the your custom command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_commandGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_commandActivateMonitorErrorCode

# verify its new value:
if { [openCarac_commandGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.13.2.4 openCarac_commandDeactivateCaseSensitivity

Sets openCarac command boolean "case sensitivity" to "0".

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac command files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractopFilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be accessed through openCarac_commandGetCaseSensitivity.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_commandDeactivateCaseSensitivity

# verify its new value:
if { [openCarac_commandGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.13.2.5 openCarac_commandDeactivateDirectoryChange

Sets openCarac command boolean "directory change" to "0".

When executing your custom command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the your custom command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be accessed through openCarac_commandGetDirectoryChange.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_commandDeactivateDirectoryChange

# verify its new value:
if { [openCarac_commandGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.13.2.6 openCarac_commandDeactivateMonitorErrorCode

Sets openCarac command boolean "monitor error code" to "0".

When executing your custom command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the your custom command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be accessed through openCarac_commandGetMonitorErrorCode.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the boolean value:
openCarac_commandDeactivateMonitorErrorCode

# verify its new value:
if { [openCarac_commandGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.13.2.7 openCarac_commandGetCaseSensitivity

Returns the value of "case sensitivity" attribute for your custom command.

When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In order to know what to substitute, this case sensitivity is used for matching. This also affects openCarac command files parser: case is not sensitive to add measures but the simulator case sensitivity is taken into account to filter devices or net names. See functions openCarac_caracGetCheckopList, openCarac_caracGetExtractopFilterList and openCarac_simulationGetExtractopList for more informations about devices or net names to be extracted. See function openCarac_runningParseSimulatorFiles for more informations about files parsing. Its value can be set through openCarac_commandActivateCaseSensitivity and openCarac_commandDeactivateCaseSensitivity.

Returns

Boolean ; 0 if "case sensitivity" attribute of your custom command is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_commandActivateCaseSensitivity

# verify its new value:
if { [openCarac_commandGetCaseSensitivity] } {
    openCarac_message "Case sensitivity is activated."
} else {
    openCarac_message "Case sensitivity is deactivated."
}
```

4.13.2.8 openCarac_commandGetCheckOptions

Returns the value of "check options" attribute for your custom command.

The your custom command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_commandSetCheckOptions.

Returns

String ; your custom command check options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_commandSetCommand        "/usr/bin/command"
openCarac_commandSetCheckOptions "-b -n"
openCarac_commandSetRunOptions     "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_commandGetCheckOptions]
} else {
    set theOptions [openCarac_commandGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_commandGetCommand] $theOptions "./mainFile.spi"} fid
```

4.13.2.9 openCarac_commandGetCommand

Returns the value of "command" attribute for your custom command.

This returns the command to execute your custom command. This command is executed by openCarac through the T↩CL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of openCarac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac_commandGetCheckOptions and openCarac_commandGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be changed through openCarac_commandGetCommand.

Returns

String ; your custom command ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_commandSetCommand        "/usr/bin/command"
openCarac_commandSetCheckOptions "-b -n"
openCarac_commandSetRunOptions     "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_commandGetCheckOptions]
} else {
    set theOptions [openCarac_commandGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_commandGetCommand] $theOptions "./mainFile.spi"} fid
```

4.13.2.10 openCarac_commandGetCommentSyntax

Returns the value of "comment syntax" attribute for your custom command.

Its value is a non-empty string that is not a list. When creating a temporary folder through openCarac_runningCreate↩TemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be set through openCarac_commandSetCommentSyntax.

Returns

String ; comment syntax, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the comment syntax:
openCarac_commandSetCommentSyntax "**"

set theComment "[openCarac_commandGetCommentSyntax] This is a comment."
puts $theComment
```

4.13.2.11 openCarac_commandGetDirectoryChange

Returns the value of "directory change" attribute for your custom command.

When executing your custom command through openCarac_runningExecuteSimulator, depending on the simulator behaviour, files inclusion can either be relative to the file they are included in or to the directory it has been executed in. If "directory change" attribute is activated, before executing the your custom command, openCarac performs a changing of directory so that files inclusion are also relative to the directory it has been executed in. After having executed the command, openCarac changes back the previous location. Its value can be set through openCarac_commandActivate↩DirectoryChange and openCarac_commandDeactivateDirectoryChange.

Returns

Boolean ; 0 if "directory change" attribute of your custom command is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_commandActivateDirectoryChange

# verify its new value:
if { [openCarac_commandGetDirectoryChange] } {
    openCarac_message "Directory change is activated."
} else {
    openCarac_message "Directory change is deactivated."
}
```

4.13.2.12 openCarac_commandGetIncDirective

Returns the value of "inc directive" attribute for your custom command.

Its value is a non-empty string that is not a list ; also, it is different from the command "lib directive" and command "param directive". To define it, case sensitivity depends on the command "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac command "lib directive" and "param directive" can be accessed through openCarac_commandGet↩LibDirective and openCarac_commandGetParamDirective. The value of openCarac command "case sensitivity" boolean can be accessed through openCarac_commandGetCaseSensitivity. Its value can be set through openCarac_command↩SetIncDirective.

Returns

String ; inc directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_commandSetIncDirective     ".INCLUDE"
openCarac_commandSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_commandGetIncDirective]
set theDelim     [openCarac_commandGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.13.2.13 openCarac_commandGetLibDirective

Returns the value of "lib directive" attribute for your custom command.

Its value is a non-empty string that is not a list ; also, it is different from the command "inc directive" and command "param directive". To define it, case sensitivity depends on the command "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of open↩ Carac command "inc directive" and "param directive" can be accessed through openCarac_commandGetIncDirective and openCarac_commandGetParamDirective. The value of openCarac command "case sensitivity" boolean can be accessed through openCarac_commandGetCaseSensitivity. Its value can be set through openCarac_commandSetLibDirective.

Returns

String ; lib directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_commandSetLibDirective     ".LIB"
openCarac_commandSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_commandGetLibDirective]
set theDelim     [openCarac_commandGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.13.2.14 openCarac_commandGetLogExtension

Returns the value of "log extension" attribute for your custom command.

This returns the log file extension to print what is returned by the your custom command in. The command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension is a lower case non-empty string, not a list itself, of at least two characters and starting with a dot (.). The value of the your custom command can be accessed through openCarac_commandGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be set through openCarac_commandSetLogExtension.

Returns

String ; log extension, single word, in lower case, of at least two characters and starting with a dot (.).

Example

```
# change the log extension:
openCarac_commandSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_commandGetCheckOptions]
} else {
    set theOptions [openCarac_commandGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_commandGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_commandGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.13.2.15 openCarac_commandGetMonitorErrorCode

Returns the value of "monitor error code" attribute for your custom command.

When executing your custom command through openCarac_runningExecuteSimulator, if custom execution mode is not activated (its value can be accessed through openCarac_applicationGetCustomExecutionMode), an error code is returned and openCarac can monitor it. If "monitor error code" attribute is activated, openCarac prints an error if the execution of the your custom command returns a non-zero error code. Otherwise, the returned error code is ignored by openCarac. Its value can be set through openCarac_commandActivateMonitorErrorCode and openCarac_commandDeactivateMonitor↩ErrorCode.

Returns

Boolean ; 0 if "monitor error code" attribute of your custom command is deactivated, 1 if it is activated.

Example

```
# change the boolean value:
openCarac_commandActivateMonitorErrorCode

# verify its new value:
if { [openCarac_commandGetMonitorErrorCode] } {
    openCarac_message "Monitor error code is activated."
} else {
    openCarac_message "Monitor error code is deactivated."
}
```

4.13.2.16 openCarac_commandGetParamDirective

Returns the value of "param directive" attribute for your custom command.

Its value must be a non-empty string that is not a list ; also, it is different from the command "inc directive" and command "lib directive". To define it, case sensitivity depends on the command "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac command "inc directive" and "lib directive" can be accessed through openCarac_commandGetIncDirective and openCarac_commandGetLibDirective. The value of openCarac command "case sensitivity" boolean can be accessed through openCarac_commandGetCase↩Sensitivity. The value of "param equality" can be accessed through openCarac_commandGetParamEquality. Its value can be set through openCarac_commandSetParamDirective.

Returns

String ; param directive, non-empty, not a list itself ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_commandSetParamDirective ".PARAM"
openCarac_commandSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_commandGetParamDirective]
set theEqual     [openCarac_commandGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.13.2.17 openCarac_commandGetParamEquality

Returns the value of "param equality" attribute for your custom command.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac command "param directive" can be accessed through openCarac_commandGetParamDirective. Its value can be set through openCarac_commandSetParamEquality.

Returns

String ; param equality ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_commandSetParamDirective ".PARAM"
openCarac_commandSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_commandGetParamDirective]
set theEqual     [openCarac_commandGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.13.2.18 openCarac_commandGetRunOptions

Returns the value of "run options" attribute for your custom command.

The your custom command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_commandSetRunOptions.

Returns

String ; your custom command run options ; integer -1 if an error occurred.

Example

```
# change the command:
openCarac_commandSetCommand      "/usr/bin/command"
openCarac_commandSetCheckOptions "-b -n"
openCarac_commandSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_commandGetCheckOptions]
} else {
    set theOptions [openCarac_commandGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_commandGetCommand] $theOptions "./mainFile.spi"} fid
```

4.13.2.19 openCarac_commandGetSaveFilter

Returns the value of "save filter" attribute for your custom command.

When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac_runningGetSimulatorFilesSavingFolderPath. Save filter is a list of strings in lower case, each of them being a single word starting with a dot (.). The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be set through openCarac_commandSetSaveFilter.

Returns

List ; strings in lower case, single words starting with a dot ; integer -1 if an error occurred.

Example

```
set theExtensionsList [openCarac_commandGetSaveFilter]

# define which files are not saved by openCarac:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*"] {

    set theExtension [string tolower [file extension $theFile]]

    if { [lsearch $theExtensionsList $theExtension] == -1 } {
        openCarac_warning "This file will not be saved by openCarac: $theFile"
    }

}
```

4.13.2.20 openCarac_commandGetStringDelimiter

Returns the value of "string delimiter" attribute for your custom command.

Its value is an empty string or a single character. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be set through openCarac_commandSetStringDelimiter.

Returns

String ; string delimiter, empty string or single character ; integer -1 if an error occurred.

Example

```
# change the syntax:
openCarac_commandSetIncDirective     ".INCLUDE"
openCarac_commandSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_commandGetIncDirective]
set theDelim     [openCarac_commandGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.13.2.21 openCarac_commandGetToRemoveInCheckMode

Returns the value of "to remove in check mode" attribute for your custom command.

This is a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and command comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the command "case sensitivity" attribute (accessible through openCarac_commandGetCaseSensitivity). The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be set through openCarac_commandSetToRemoveInCheckMode.

Returns

List ; Strings that are not empty and not lists themselves ; integer -1 if an error occurred.

Example

```
# set the list of directives to remove:
openCarac_commandSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_commandGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

4.13.2.22 openCarac_commandSetCheckOptions *value*

Sets the value of "check options" attribute for your custom command.

The your custom command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is activated, the command is concatenated with the value of this "check options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_commandGetCheckOptions.

Parameters

value	: String ; your custom command check options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_commandSetCommand      "/usr/bin/command"
openCarac_commandSetCheckOptions "-b -n"
openCarac_commandSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_commandGetCheckOptions]
} else {
    set theOptions [openCarac_commandGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_commandGetCommand] $theOptions "./mainFile.spi"} fid
```

4.13.2.23 openCarac_commandSetCommand *value*

Sets the value of "command" attribute for your custom command.

This sets the command to execute your custom command. This command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. It is concatenated with either "run options" or "check options" depending on the value of openCarac *application* "check mode" boolean. For more informations about "run options" or "check options", see access functions openCarac_commandGetCheckOptions and openCarac_commandGetRunOptions. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. Its value can be accessed through openCarac_commandGetCommand.

Parameters

value	: String ; your custom command.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_commandSetCommand      "/usr/bin/command"
openCarac_commandSetCheckOptions "-b -n"
openCarac_commandSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_commandGetCheckOptions]
} else {
    set theOptions [openCarac_commandGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_commandGetCommand] $theOptions "./mainFile.spi"} fid
```

4.13.2.24 openCarac_commandSetCommentSyntax *value*

Sets the value of "comment syntax" attribute for your custom command.

Its value must be a non-empty string that is not a list. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied and substitutions occur. In case a line must be removed by openCarac, the "comment syntax" is added at the beginning of the line. Its value can be accessed through openCarac_commandGetCommentSyntax.

Parameters

value	: String ; comment syntax, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the comment syntax:
openCarac_commandSetCommentSyntax "**"

set theComment "[openCarac_commandGetCommentSyntax] This is a comment."
puts $theComment
```

4.13.2.25 openCarac_commandSetIncDirective *value*

Sets the value of "inc directive" attribute for your custom command.

Its value must be a non-empty string that is not a list ; also, it must be different from the command "lib directive" and command "param directive". To define it, case sensitivity depends on the command "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "inc directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and openCarac *simulation* or netlist files are included. To substitute, openCarac considers that a line matches a file inclusion when it starts with this "inc directive". To include an openCarac *simulation* or netlist file, this "inc directive" is added at the beginning of the line. The values of openCarac command "lib directive" and "param directive" can be accessed through openCarac_commandGetLibDirective and openCarac_commandGetParamDirective. The value of openCarac command "case sensitivity" boolean can be accessed through openCarac_commandGetCaseSensitivity. Its value can be accessed through openCarac_commandGetIncDirective.

Parameters

value	: String ; inc directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_commandSetIncDirective     ".INCLUDE"
openCarac_commandSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_commandGetIncDirective]
set theDelim     [openCarac_commandGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.13.2.26 openCarac_commandSetLibDirective *value*

Sets the value of "lib directive" attribute for your custom command.

Its value must be a non-empty string that is not a list ; also, it must be different from the command "inc directive" and command "param directive". To define it, case sensitivity depends on the command "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "lib directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and model or libparam files are loaded. To substitute, openCarac considers that a line matches a file loading when it starts with this "lib directive". To load a model or libparam file, this "lib directive" is added at the beginning of the line. The values of openCarac command "inc directive" and "param directive" can be accessed through openCarac_commandGetIncDirective and openCarac_commandGetParamDirective. The value of openCarac command "case sensitivity" boolean can be accessed through openCarac_commandGetCaseSensitivity. Its value can be accessed through openCarac_commandGetLibDirective.

Parameters

value	: String ; lib directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_commandSetLibDirective     ".LIB"
openCarac_commandSetStringDelimiter "\""

# loading of a library:
set theDirective [openCarac_commandGetLibDirective]
set theDelim     [openCarac_commandGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.lib$theDelim THE_LIB_NAME"
puts $theInclusion
```

4.13.2.27 openCarac_commandSetLogExtension *value*

Sets the value of "log extension" attribute for your custom command.

This sets the log file extension to print what is returned by the your custom command in. The command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. What has been printed by the command is caught by openCarac and written in a file having the same root name as the main file in the temporary folder and this "log extension". Log file extension must be a non-empty string, not a list itself, of at least two characters and starting with a dot (.), openCarac automatically converts it to lower case. If the log file extension does not appear in the "save filter" attribute of openCarac your custom command, accessible through openCarac_commandGetSaveFilter, it is automatically added. The value of the your custom command can be accessed through openCarac_commandGetCommand. The value of the main file in the temporary folder can be accessed through openCarac_runningGetFromMainFilePath. Its value can be accessed through openCarac_commandGetLogExtension.

Parameters

value	: String ; log extension, single word, of at least two characters and starting with a dot (.).

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the log extension:
openCarac_commandSetLogExtension ".log"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_commandGetCheckOptions]
} else {
    set theOptions [openCarac_commandGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_commandGetCommand] $theOptions "./mainFile.spi"} fid

# print the output in the log file:
set theLogFile "[file rootname "./mainFile.spi"][openCarac_commandGetLogExtension]"
set buf [open $theLogFile a]
puts $buf $fid
close $buf
```

4.13.2.28 openCarac_commandSetParamDirective *value*

Sets the value of "param directive" attribute for your custom command.

Its value must be a non-empty string that is not a list ; also, it must be different from the command "inc directive" and command "lib directive". To define it, case sensitivity depends on the command "case sensitivity" boolean attribute. It is expecting a syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with this "param directive" and that the "param equality" is located between its name and its value. To set a parameter that has not been found in the files, this "param directive" is added at the beginning of the line. The values of openCarac command "inc directive" and "lib directive" can be accessed through openCarac_commandGetIncDirective and openCarac_commandGetLibDirective. The value of openCarac command "case sensitivity" boolean can be accessed through openCarac_commandGetCaseSensitivity. The value of "param equality" can be accessed through openCarac_commandGetParamEquality. Its value can be accessed through openCarac_commandGetParamDirective.

Parameters

value	: String ; param directive, non-empty, not a list itself.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_commandSetParamDirective ".PARAM"
openCarac_commandSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_commandGetParamDirective]
set theEqual     [openCarac_commandGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.13.2.29 openCarac_commandSetParamEquality *value*

Sets the value of "param equality" attribute for your custom command.

It is the string located between a parameter name and its value to match the syntax based on the Spice "param directive" to define how substitutions are performed by openCarac. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and parameters values are tuned. To substitute, openCarac considers that a line matches a parameter setting when it starts with the "param directive" and this "param equality" is located between the parameter name and its value. To set a parameter that has not been found in the files, this "param equality" is added between the parameter name and its value. The value of openCarac command "param directive" can be accessed through openCarac_commandGetParamDirective. Its value can be accessed through openCarac_commandGetParamEquality.

Parameters

value	: String ; param equality.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_commandSetParamDirective ".PARAM"
openCarac_commandSetParamEquality  "="

set theName  "myParam"
set thevalue "42"

# setting of a parameter:
set theDirective [openCarac_commandGetParamDirective]
set theEqual     [openCarac_commandGetParamEquality]
set theParameterSetting "$theDirective $theName $theEqual $thevalue"
puts $theParameterSetting
```

4.13.2.30 openCarac_commandSetRunOptions *value*

Sets the value of "run options" attribute for your custom command.

The your custom command is executed by openCarac through the TCL exec command when calling openCarac_runningExecuteSimulator if openCarac *application* "custom execution mode" boolean is not activated. If openCarac *application* "check mode" boolean is deactivated, the command is concatenated with the value of this "run options" attribute. The value of openCarac *application* "custom execution mode" boolean can be accessed through openCarac_applicationGetCustomExecutionMode. The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_commandGetRunOptions.

Parameters

value	: String ; your custom command run options.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the command:
openCarac_commandSetCommand      "/usr/bin/command"
openCarac_commandSetCheckOptions "-b -n"
openCarac_commandSetRunOptions   "-b"

# select the options:
if { [openCarac_applicationGetCheckMode] } {
    set theOptions [openCarac_commandGetCheckOptions]
} else {
    set theOptions [openCarac_commandGetRunOptions]
}

# execute the simulator:
catch { eval exec -- [openCarac_commandGetCommand] $theOptions "./mainFile.spi"} fid
```

4.13.2.31 openCarac_commandSetSaveFilter *value*

Sets the value of "save filter" attribute for your custom command.

This must be a list of strings, each of them being a single word starting with a dot (.), openCarac automatically converts them to lower case. When calling openCarac files parser through openCarac_runningParseSimulatorFiles, if openCarac *application* "simulator files copy" boolean is activated, a copy of files having their extension matching a pattern in this list is performed from the temporary folder into the directory defined by the "simulator files saving folder path". Matching follows the rules of TCL "string match" command without case-sensitivity. The destination folder path can be accessed through openCarac_runningGetSimulatorFilesSavingFolderPath. The value of openCarac *application* "simulator files copy" attribute can be accessed through openCarac_applicationGetSimulatorFilesCopy. Its value can be accessed through openCarac_commandGetSaveFilter.

Parameters

value	: List ; strings, single words starting with a dot.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
set theFilesExtensionsFilter [openCarac_applicationGetFilesExtensionFilter]
set theSaveFilterList        [list]

# use the extensions of the files in the current directory:
foreach theFile [glob -nocomplain -directory [pwd] -type {f} "*.*"] {

    set theExtension [string tolower [file extension $theFile]]

    # ignore the files that have been copied by openCarac:
    if { [lsearch $theFilesExtensionsFilter $theExtension] == -1 } {
        continue
    }

    if { [lsearch $theSaveFilterList $theExtension] == -1 } {
        lappend theSaveFilterList $theExtension
    }

}

# apply this filter to openCarac:
openCarac_commandSetSaveFilter $theSaveFilterList
```

4.13.2.32 openCarac_commandSetStringDelimiter *value*

Sets the value of "string delimiter" attribute for your custom command.

Its value must be an empty string or a single character. When creating a temporary folder through openCarac_runningCreateTemporaryFolder, files are copied, substitutions occur and files are included or loaded. In each case of path substitution, simulator "string delimiter" is used before and after the path addition. Its value can be accessed through openCarac_commandGetStringDelimiter.

Parameters

value	: String ; string delimiter, empty string or single character.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# change the syntax:
openCarac_commandSetIncDirective     ".INCLUDE"
openCarac_commandSetStringDelimiter "\""

# inclusion of a file:
set theDirective [openCarac_commandGetIncDirective]
set theDelim     [openCarac_commandGetStringDelimiter]
set theInclusion "$theDirective $theDelim../myFile.inc$theDelim"
puts $theInclusion
```

4.13.2.33 openCarac_commandSetToRemoveInCheckMode *value*

Sets the value of "to remove in check mode" attribute for your custom command.

This must be a list of patterns that are not lists themselves and are not empty strings. When having openCarac *application* "check mode" boolean activated, openCarac aims to quickly verify that no error would occur when executing the simulator. To make sure that a simulator check does not take too much time, some lines from the files to copy in the temporary folders can be removed. When calling openCarac_runningCreateTemporaryFolder, files are copied and, if openCarac *application* "check mode" boolean is activated, any line starting with a pattern from this list is substituted and command comment syntax is added at the beginning of the line. Matching follows the rules of TCL "string equal" command ; case sensitivity depends on the command "case sensitivity" attribute (accessible through openCarac_commandGetCaseSensitivity). The value of openCarac *application* "check mode" boolean can be accessed through openCarac_applicationGetCheckMode. Its value can be accessed through openCarac_commandGetToRemoveInCheckMode.

Parameters

value	: List ; Strings that are not empty and not lists themselves.

Returns

Integer ; -1 if an error occurred, 0 otherwise.

Example

```
# set the list of directives to remove:
openCarac_commandSetToRemoveInCheckMode [list ".TRAN" ".AC" ".DC" ".NOISE"]

# the list is not empty:
foreach theDirective [openCarac_commandGetToRemoveInCheckMode] {
    puts "Lines starting with \"$theDirective\" are removed in check mode."
}
```

Chapter 5

File Documentation

5.1 customProcedures.tcl File Reference

Here are defined procedures used by openCarac, this file is available in openCarac sources and in the user home directory. It allows the user to run his simulator the way he wants to and customize the behaviour of openCarac execution.

Functions

- openCarac_customRunSimulator theMainFilePath

  Executes the simulator in a cusom way.
- openCaracHook_ON_PRE_APPLICATION_LOAD_ENVIRONMENT

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_APPLICATION_LOAD_ENVIRONMENT

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_APPLICATION_CREATE_FULL_SUMMARY_FILE

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_APPLICATION_CREATE_FULL_SUMMARY_FILE

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_CARAC_MAKE_READY_FOR_RUNNINGS theCarac

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_CARAC_MAKE_READY_FOR_RUNNINGS theCarac

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_CARAC_EXTRACT_RESULTS theCarac

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_CARAC_EXTRACT_RESULTS theCarac

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_CONFIGURATION_OPEN theConfigurationFilePath

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_CONFIGURATION_OPEN theConfigurationFilePath

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_CONFIGURATION_CREATE_ARCHIVES theConfiguration

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_CONFIGURATION_CREATE_ARCHIVES theConfiguration

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_RUNNING_CREATE_TEMPORARY_FOLDER theRunning

Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_RUNNING_CREATE_TEMPORARY_FOLDER theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_RUNNING_EXECUTE_SIMULATOR theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_RUNNING_EXECUTE_SIMULATOR theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_RUNNING_PARSE_SIMULATOR_FILES theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_RUNNING_PARSE_SIMULATOR_FILES theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_RUNNING_DELETE_TEMPORARY_FOLDER theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_RUNNING_DELETE_TEMPORARY_FOLDER theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_RUNNING_SAVE_RESULTS theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_RUNNING_SAVE_RESULTS theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_RUNNING_EXTRACT_RESULTS theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_POST_RUNNING_EXTRACT_RESULTS theRunning

  Hook to execute some code at a given instant in openCarac execution.
- openCaracHook_ON_PRE_EXIT

  Hook to execute some code at a given instant in openCarac execution.

5.1.1 Detailed Description

Here are defined procedures used by openCarac, this file is available in openCarac sources and in the user home directory. It allows the user to run his simulator the way he wants to and customize the behaviour of openCarac execution.

The user may like to run his simulator through another program or on another computer. He can use custom procedures to do so; althrough, those conditions may run the simulator in background and make openCarac remove the temporary folder and extract the results before the end of the simulations. To avoid this kind of inconvenient, hook procedures can also be defined: openCarac execution pauses until these procedures end.

customProcedures file must be located next to openCarac main executable and into the user home directory (after a configuration, i.e. calling openCarac_applicationParseArgv with --configure option). The default customProcedures file is used by openCarac until openCarac_applicationLoadEnvironment is called ; then the user defined customProcedures file overloads the functions definition.

5.1.2 Function Documentation

5.1.2.1 openCarac_customRunSimulator *theMainFilePath*

Executes the simulator in a cusom way.

It uses its main file path and API functions to define how to execute the appropriate simulator. By default, this function:

- accesses the openCarac *running* from its main file path through openCarac_runningGetFromMainFilePath
- gets the parent openCarac *simulation* and parent openCarac *carac* with openCarac_runningGetParentSimulation and openCarac_simulationGetParentCarac.

- defines the selected simulator through openCarac_caracGetSimulator.
- depending on the simulator, it defines its "command", "check options", "run options" and "log extension". (this can be done through openCarac_ngspiceGetCommand, openCarac_ngspiceGetCheckOptions, openCarac_ngspiceGet↩RunOptions and openCarac_ngspiceGetLogExtension for ngspice)
- selects either "check options" or "run options" depending on the value of "check mode", accessible through open↩Carac_applicationGetCheckMode.
- executes the command with the options and the "main file path" as arguments ; catches what is returned.
- prints what is returned into a file with the same root name as the "main file path" and the "log extension".

Note that, to make sure that openCarac properly parses the simulator files when calling openCarac_runningParse↩SimulatorFiles, the log file must have the appropriate extension.

Parameters

theMainFilePath	String ; absolute path to the main file that must be loaded with the simulator.

Returns

Can return anything, it remains unused by openCarac.

5.1.2.2 openCaracHook_ON_POST_APPLICATION_CREATE_FULL_SUMMARY_FILE

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_applicationCreateFullSummaryFile.

5.1.2.3 openCaracHook_ON_POST_APPLICATION_LOAD_ENVIRONMENT

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_applicationLoadEnvironment.

5.1.2.4 openCaracHook_ON_POST_CARAC_EXTRACT_RESULTS *theCarac*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_caracExtractResults.

Parameters

theCarac	The argument that is given to openCarac_caracExtractResults.

5.1.2.5 openCaracHook_ON_POST_CARAC_MAKE_READY_FOR_RUNNINGS *theCarac*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_caracMakeReadyForRunnings.

Parameters

theCarac	The argument that is given to openCarac_caracMakeReadyForRunnings.

5.1.2.6 openCaracHook_ON_POST_CONFIGURATION_CREATE_ARCHIVES *theConfiguration*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_configurationCreateArchives.

Parameters

theConfiguration	The argument that is given to openCarac_configurationCreateArchives.

5.1.2.7 openCaracHook_ON_POST_CONFIGURATION_OPEN *theConfigurationFilePath*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_configurationOpen.

Parameters

the↩ Configuration↩ FilePath	The argument that is given to openCarac_configurationOpen.

5.1.2.8 openCaracHook_ON_POST_RUNNING_CREATE_TEMPORARY_FOLDER *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_runningCreateTemporaryFolder.

Parameters

theRunning	The argument that is given to openCarac_runningCreateTemporaryFolder.

5.1.2.9 openCaracHook_ON_POST_RUNNING_DELETE_TEMPORARY_FOLDER *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_runningDeleteTemporaryFolder.

Parameters

theRunning	The argument that is given to openCarac_runningDeleteTemporaryFolder.

5.1.2.10 openCaracHook_ON_POST_RUNNING_EXECUTE_SIMULATOR *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_runningExecuteSimulator.

Parameters

theRunning	The argument that is given to openCarac_runningExecuteSimulator.

5.1.2.11 openCaracHook_ON_POST_RUNNING_EXTRACT_RESULTS *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_runningExtractResults.

Parameters

theRunning	The argument that is given to openCarac_runningExtractResults.

5.1.2.12 openCaracHook_ON_POST_RUNNING_PARSE_SIMULATOR_FILES *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_runningParseSimulatorFiles.

Parameters

theRunning	The argument that is given to openCarac_runningParseSimulatorFiles.

5.1.2.13 openCaracHook_ON_POST_RUNNING_SAVE_RESULTS *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the end of openCarac_runningSaveResults.

Parameters

theRunning	The argument that is given to openCarac_runningSaveResults.

5.1.2.14 openCaracHook_ON_PRE_APPLICATION_CREATE_FULL_SUMMARY_FILE

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_applicationCreateFullSummaryFile.

5.1.2.15 openCaracHook_ON_PRE_APPLICATION_LOAD_ENVIRONMENT

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_applicationLoadEnvironment.

5.1.2.16 openCaracHook_ON_PRE_CARAC_EXTRACT_RESULTS *theCarac*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_caracExtractResults.

Parameters

theCarac	The argument that is given to openCarac_caracExtractResults.

5.1.2.17 openCaracHook_ON_PRE_CARAC_MAKE_READY_FOR_RUNNINGS *theCarac*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_caracMakeReadyForRunnings.

Parameters

theCarac	The argument that is given to openCarac_caracMakeReadyForRunnings.

5.1.2.18 openCaracHook_ON_PRE_CONFIGURATION_CREATE_ARCHIVES *theConfiguration*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_configurationCreateArchives.

Parameters

theConfiguration	The argument that is given to openCarac_configurationCreateArchives.

5.1.2.19 openCaracHook_ON_PRE_CONFIGURATION_OPEN *theConfigurationFilePath*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_configurationOpen.

Parameters

the↩ Configuration↩ FilePath	The argument that is given to openCarac_configurationOpen.

5.1.2.20 openCaracHook_ON_PRE_EXIT

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_exit.

5.1.2.21 openCaracHook_ON_PRE_RUNNING_CREATE_TEMPORARY_FOLDER *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_runningCreateTemporaryFolder.

Parameters

theRunning	The argument that is given to openCarac_runningCreateTemporaryFolder.

5.1.2.22 openCaracHook_ON_PRE_RUNNING_DELETE_TEMPORARY_FOLDER *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_runningDeleteTemporaryFolder.

Parameters

theRunning	The argument that is given to openCarac_runningDeleteTemporaryFolder.

5.1.2.23 openCaracHook_ON_PRE_RUNNING_EXECUTE_SIMULATOR *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_runningExecuteSimulator.

Parameters

theRunning	The argument that is given to openCarac_runningExecuteSimulator.

5.1.2.24 openCaracHook_ON_PRE_RUNNING_EXTRACT_RESULTS *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_runningExtractResults.

Parameters

theRunning	The argument that is given to openCarac_runningExtractResults.

5.1.2.25 openCaracHook_ON_PRE_RUNNING_PARSE_SIMULATOR_FILES *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_runningParseSimulatorFiles.

Parameters

theRunning	The argument that is given to openCarac_runningParseSimulatorFiles.

5.1.2.26 openCaracHook_ON_PRE_RUNNING_SAVE_RESULTS *theRunning*

Hook to execute some code at a given instant in openCarac execution.

The hooks are executed before or after executing the main code of an openCarac API function. This hook is executed at the beginning of openCarac_runningSaveResults.

Parameters

theRunning	The argument that is given to openCarac_runningSaveResults.

Index