User Guide:
Runtime System Configurator

Document Version 1.0

tech_doc_e.doc / V1.2

CONTENT

1 Purpose of the RtsConfigurator

RtsConfigurator signifies „Runtimesystem Configurator“. The RtsConfigurator can be used by the developper to configure a runtime system concerning its components. In doing so the tool detects any dependencies between the particular components and allows the user to resolve these dependencies.

The RtsConfigurator can save the currently defined configuration of the runtime system in a file. In this case only those data will be stored, which are needed to restore the configuration, i.e. no data contained in the components or components themselves will be stored.

After a correct configuration the RtsConfigurator can create various files which are needed or useful to set up the runtime system. The currently supported output formats are described in 4.3.1 'All output files'.

2 First Startup and Configuration via Wizard

2.1 First startup

After the first startup the RtsConfigurator appears in its initial state.

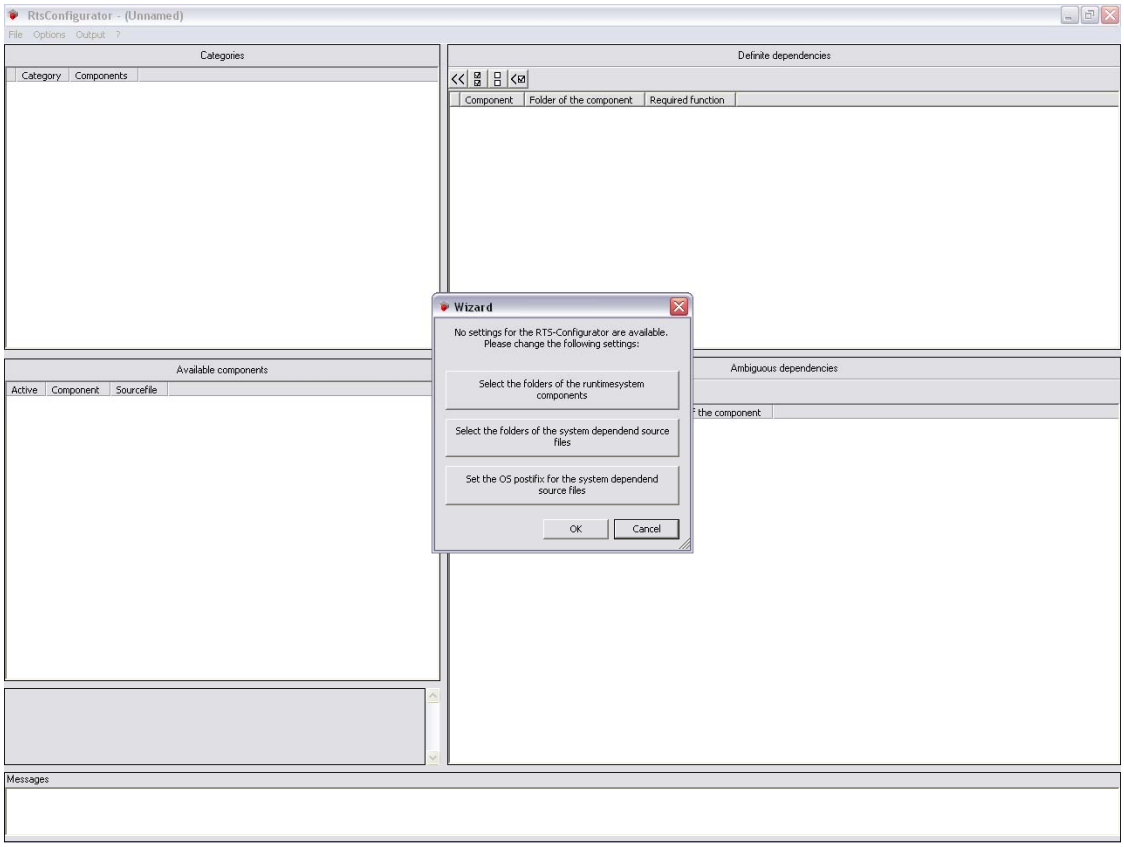


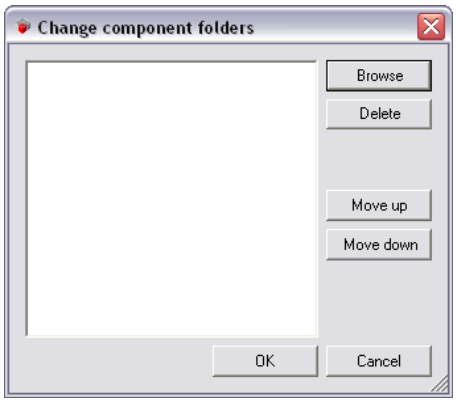
Fig.1: RtsConfigurator after initial startup

In this state there are not yet any components available for configuration. So a wizard will open where you can do the necessary settings.

2.2 Settings via the Wizard

2.2.1 Runtime System Components

In order to get displayed any components, use button **Select the folder of your runtimesystem components**. The following dialog will appear:



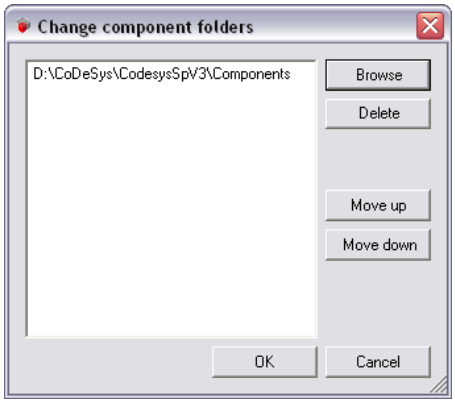


Fig.2: Dialog for defining the component folders to be browsed

Due to the fact that the component folders will be browsed recursively, the superior path can be used as source path. Via button **Browse** the components folders' paths can be selected in a dialog. Each path which is displayed in the list can be edited there. After having selected all paths/folders the settings are to be confirmed with **OK** and you will return to the wizard.

2.2.2 System dependent source files

In some components there are system-dependent source files. The storage path of these components is to be configured separately from the components. After having activated button **Select the folders of the system dependent source files** the following dialog opens.

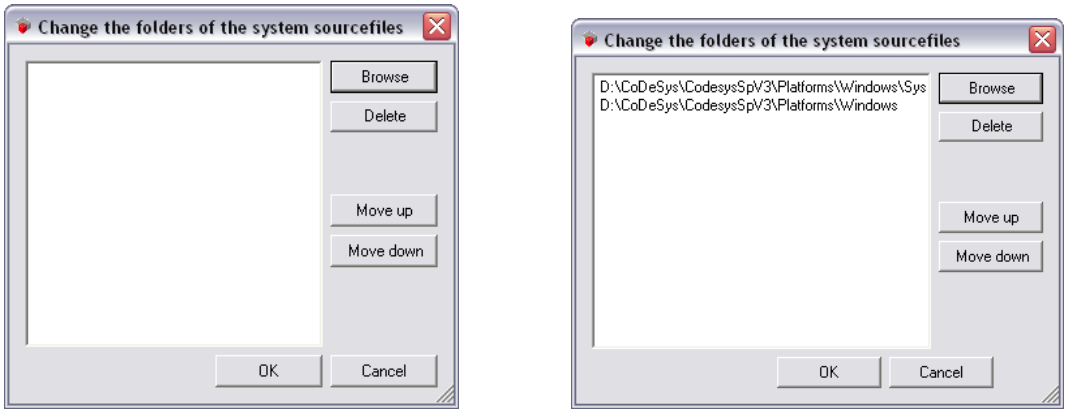


Fig.3: Dialog for setting up the paths of the sourcefiles to be browsed

The dialog is to be used like that which is available for selecting the component paths described in chap. 2.2.1, Runtime System Components. The difference is that the paths for the system dependent source files do **not get browsed recursively**. After having selected the desired folders the settings must be confirmed with **OK** and you will return to the wizard.

2.2.3 OS Postfix

The OS Postfix is needed for finding the source files which are required for a specified system. To configure the OS Postfix use button **Set the OS postfix for the system dependent source files**. The following dialog opens:

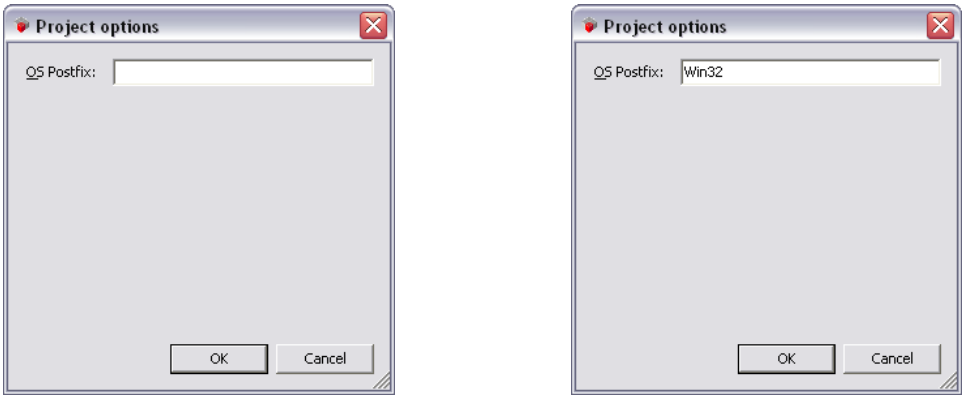


Fig.4 Project Options for setting up the OS Postfix

After having entered the OS Postfix close the dialog with **OK** and terminate the wizard with **OK**. After that the RtsConfigurator starts browsing the component paths for *.m4-files, which are describing the components. This might need some time, depending on the system and the currently configured browse-paths.

2.3 Selection of the desired Components

After having configured the component paths and the OS Postfix appropriately, several entries should be displayed in the lists **Categories** and **Available components** in the left part of the window.

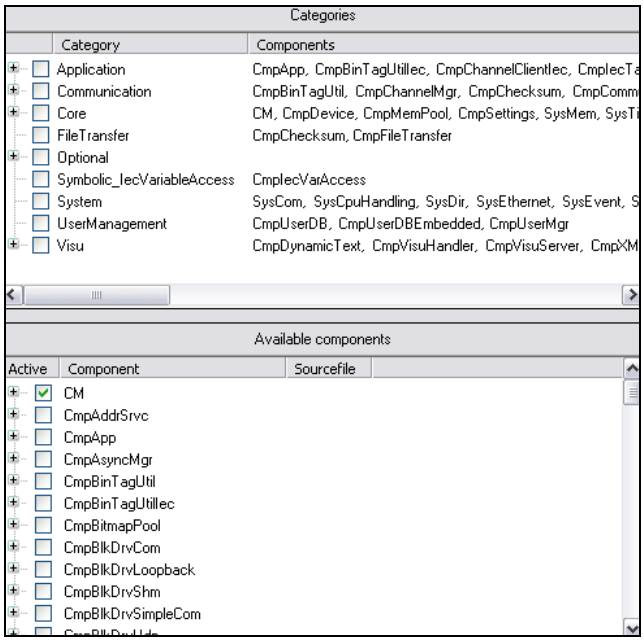


Fig.5 Dialog ‚Categories' (for selecting the components)

Via the checkbox in the first column the desired components can be activated resp. the non-desired can be deactivated. Notice that component **CM** (ComponentManager) is always active and cannot be deactivated.

See chap. 5, Errors and Warnings, for some information on how to fix any errors, which are reported in the **Messages** window.

2.4 Creating the first output

If no more errors are detected in the configuration, the first output can be triggered. Use command 'Output' – 'All output files'.
Before the output files can be created, the configuration must be saved via the „Save“- dialog which will open automatically. Notice that the output files will get stored parallel to the configuration file. An arbitrary location and file name for the configuration file can be specified and the configuration be saved. Immediately after having saved the configuration also the output files will be created. Which output files will be produced is described in chap. 4.3.1 'All output files'.

3 Main Window

The main window of the RtsConfigurator consists of several functional areas. These areas are numbered in the following figure and described in the following.

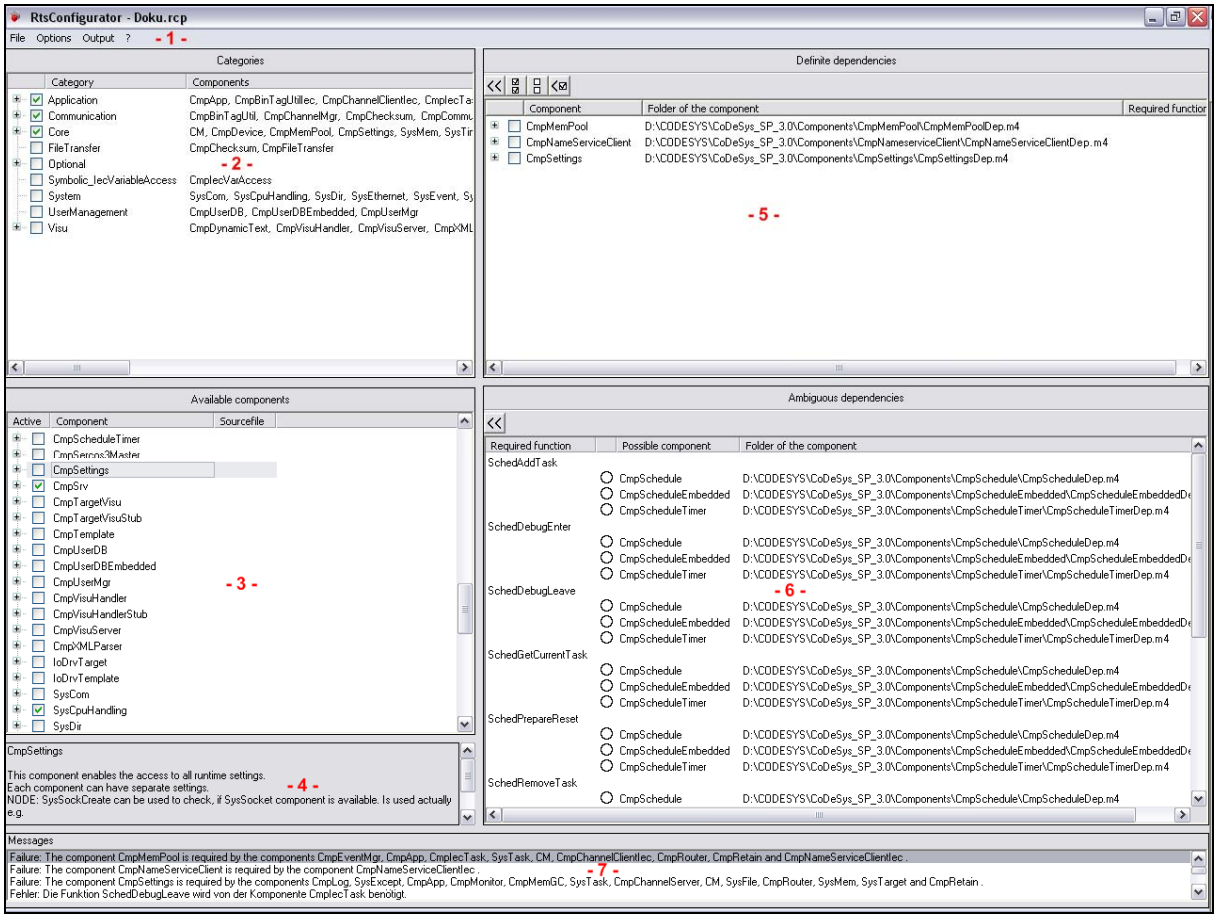


Fig.6 Main window

3.1 Menu Bar

The menu bar (- 1 -) contains commands for creating, loading and saving configuration projects, for the configuration of paths and for creating the particular output files.

Fig.7 Menu bar

3.2 Categories

In area **Categories** (- 2 -) all available system components are sorted by categories. The categories are structured hierarchically. Multiple components can be assigned to one category and a component can be available in multiple categories.

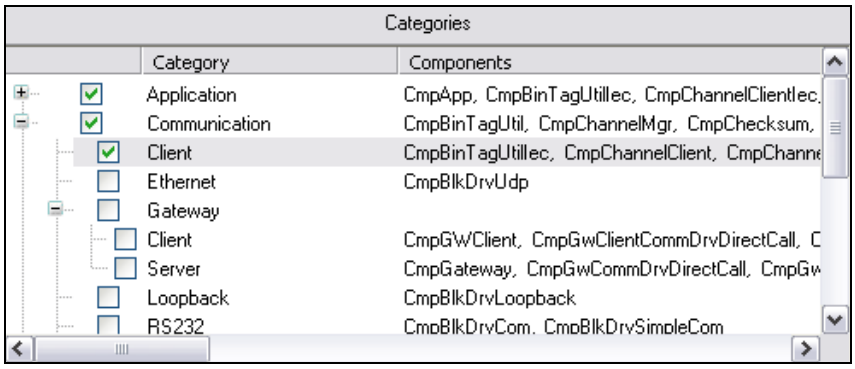


Fig.8 Categories and assigned components

A category can be activated or deactivated via the preceded checkbox.

When activating a category, all components of that category will be activated. If a sub-category is activated, the superior categories will be activated too.

When deactivating a category, the components of that category will only be deactivated if they are not also represented in another active category. A category can only be deactivated if no sub-category is still activated.

3.3 Available components

The list **Available components** (- 3 -) shows all components which have been detected in the currently configured component paths. For each component the current state (i.e. activated or deactivated) and the source files will be displayed. Via the preceded checkbox each component can be activated or deactivated. If a component is displayed in expanded mode, the source files of the component will also be shown. The files are searched automatically and can be configured after a double-click on the file name.

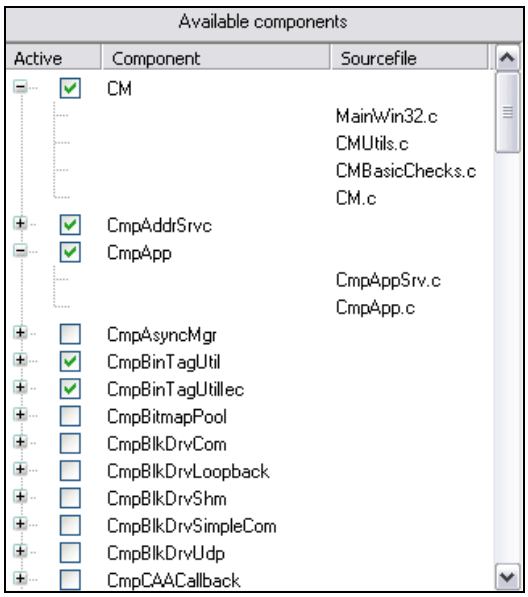


Fig.9 Available components

3.4 Component Description

This area (- 4 -) shows a description for the component currently selected in the list of "Available components".

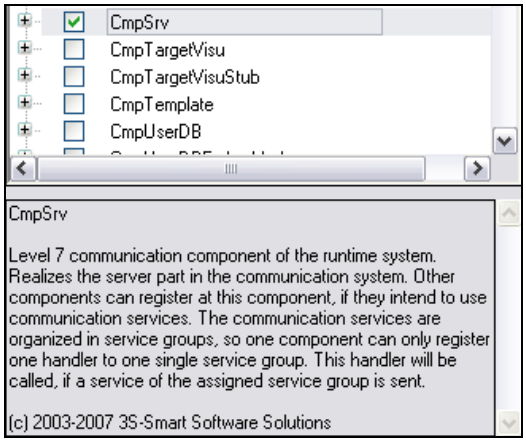


Fig.10 Component description

3.5 Definite dependencies

The list of "Definite dependencies" (- 5 -) shows unique dependencies between components. Each component listed here implements a function or several functions, which are needed by another active component. After expanding the nodes these required functions will be displayed.

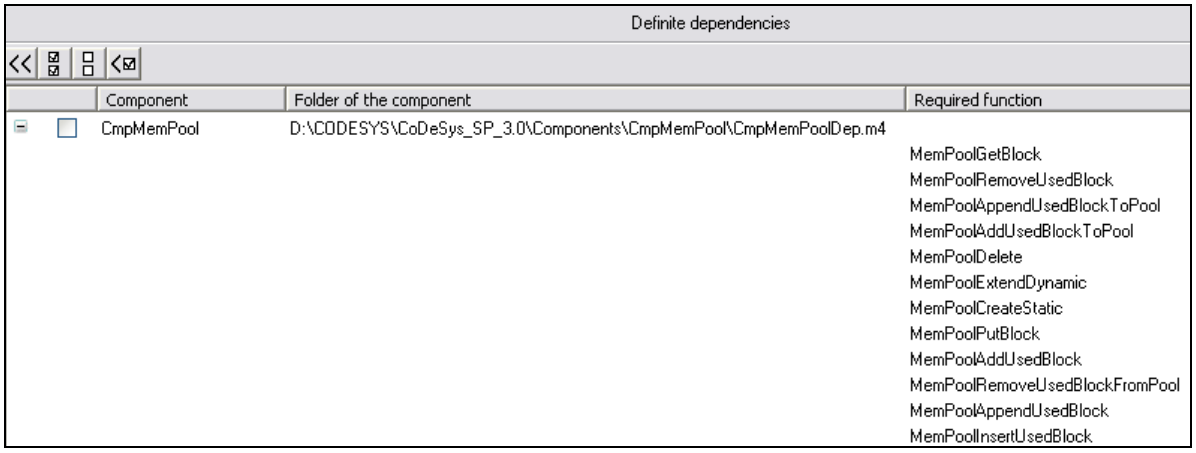


Fig.11 Definite dependencies

The function bar for the "Definite dependencies" list contains the following commands:

- **Activate selected components** , Symbol: <<
  All components selected in the list will be activated.
- **Select all components** , Symbol: ☑
  All components in the list will be selected.
- **Deselect all components**, Symbol: ☐
  All components in the list will be deselected.
- **Activate all required components** , Symbol: <☑
  All components in the list will be selected and activated until no more will be left.

3.6 Ambiguous dependencies

The "Ambiguous dependencies" list shows all functions which are needed by active components. Due to the fact that the functions are provided by multiple components, for each particular function all components implementing this function get listed.

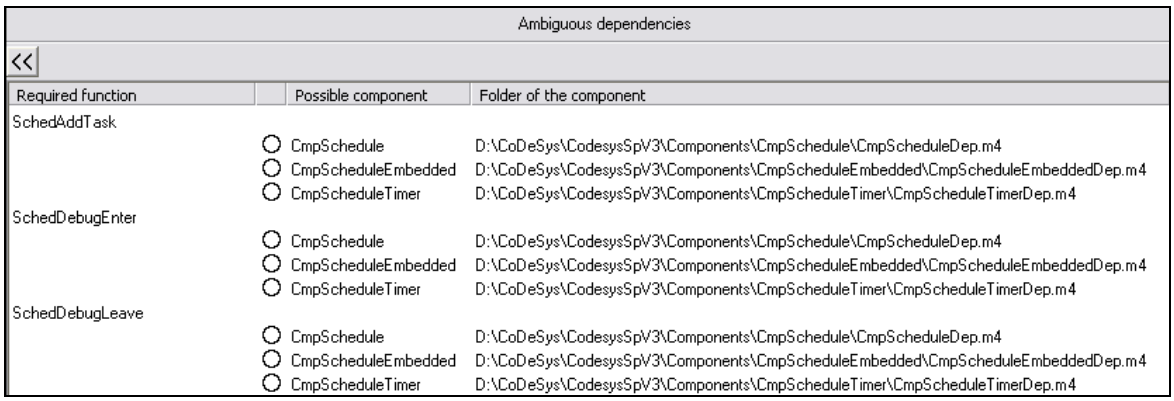


Fig.12 Non-ambiguous dependencies

Via a mouse-click on the preceded round box a component can be selected. If a component gets selected, it will be selected also in all other functions where it is detected.

The function bar for the "Ambiguous dependencies" list contains the following function:

Activate selected components , Symbol: <<
All components selected in the list will be activated.

3.7 Message window

The message window is positioned in the lower part of the main window. It shows all error messages and warnings referring to the configuration of the runtime system.

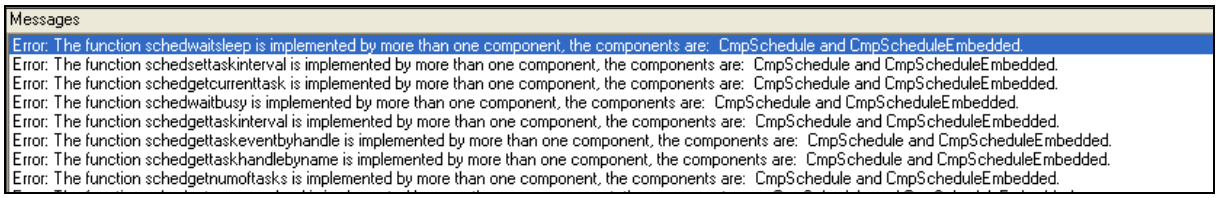


Fig.13 Message window

3.8 Screen dividers

A screen divider separates two non-overlapping windows within the user interface. See in the following figure all possible screen dividers marked by red color.

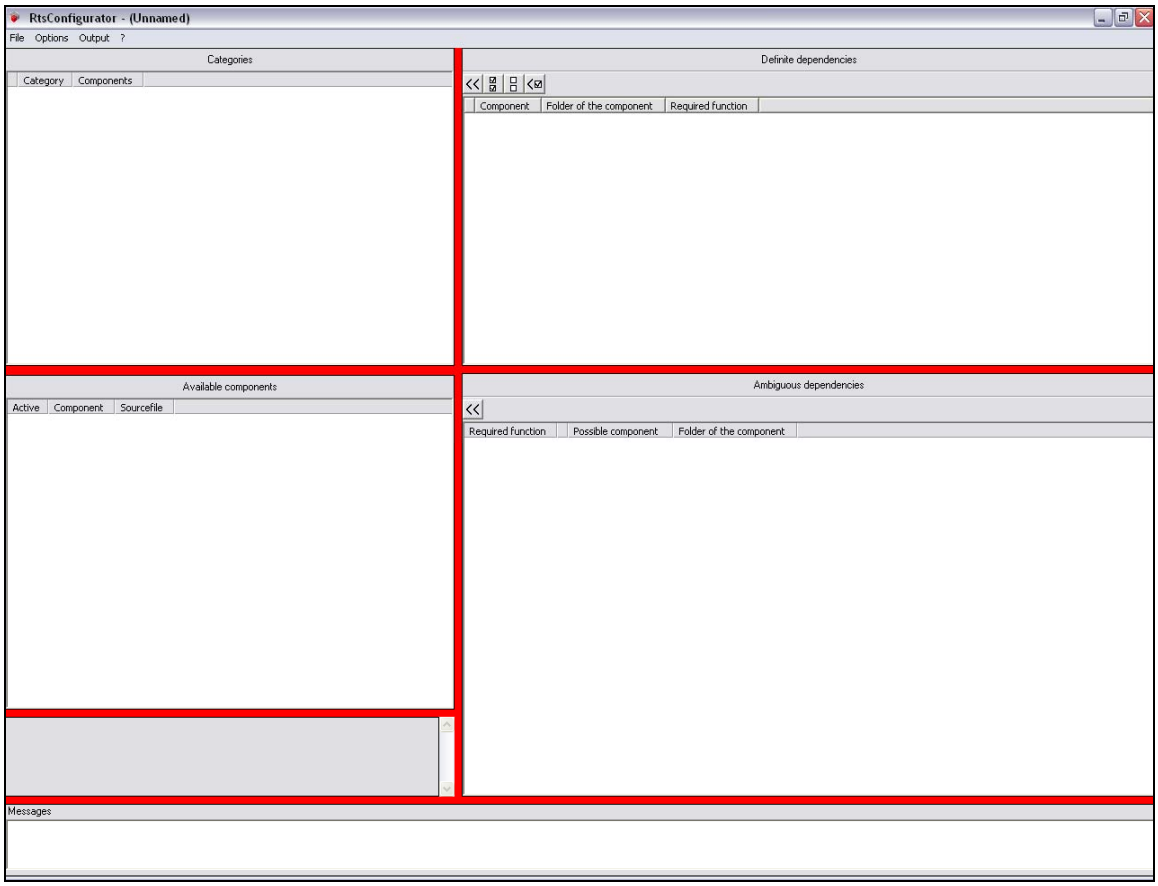


Fig.14 Screen dividers

If you place the mouse pointer on a screen divider, you can shift the divider by moving the mouse while keeping the left mouse button pressed.

4 Function Overview

4.1 File

4.1.1 'New'

This command creates a new configuration project with name “Unnamed’. This name must be changed when saving the project.

4.1.2 'Open…'

This command opens an existing configuration project. The dialog for opening a file will appear and you must select an appropriate project file with extension „*.rcp“.

4.1.3 'Save'

This command saves the project with the file name which is displayed in the title bar..

If the current file name is “(Unnamed)*“, you must specify another name (see 4.1.4 'Save As…). The default extension for a project file is „*.rcp“ .

4.1.4 'Save As…'

This command saves the current project in another resp. new file. If an already existing project gets saved in this way to a new file, the original project will not be modified.

After having selected the command, the dialog for saving a file will open. Either choose an existing file name or enter a new on. The file name will get the extension „.rcp“. If you have chosen the name of an already existing file, after having pressed button „Save“ you will get asked whether you want to overwrite this existing file.

4.1.5 'Description…'

This command can be used to modify the description for the current project. The dialog for modifying the project description will open and you can edit resp. enter any desired description in the text field.

After having done all entries, press **OK** and the description will be applied to the current project.

4.1.6 'Quit'

This command terminates the RtsConfigurator.

If the project has been modified but not yet saved, you will be asked whether the modified project should be saved or not. If the project still is “Unnamed", you will be asked to define a name (see 4.1.4 'Save As…).

4.2 Options

All commands referring to general options are available in the ‚Options' menu.

4.2.1 'Component folders…'

This command opens the dialog for selecting a path. The RtsConfigurator will recursively browse all paths displayed in the list for available components. To add a component to the ‚Available components' list, enter the path of the component files to the list.

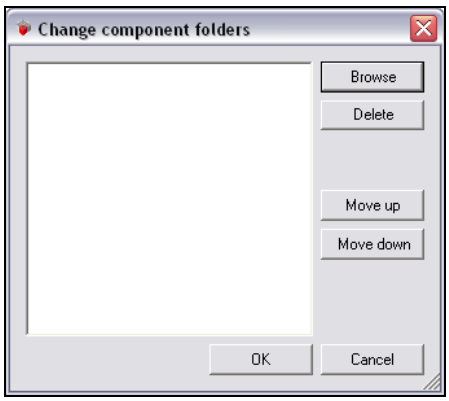


Fig.15 Change Component folders

Via button **Browse** you can open a dialog for selecting the desired path. After having closed this dialog with OK, the chosen path will be added to the list and can be edited there.

Via button **Delete** an entries can be removed from the list. The order of the path entries within the list is without any effect but it can be modified by selecting an entry and using the buttons **Move up** resp. **Move down**.

4.2.2 'Folder of system components…'

This command opens the dialog for selecting a path. The RtsConfigurator will browse all listed paths for the system-dependent source files of the components.

On a development system various versions of a system-dependent source file might be available, e.g. one version for Linux and one for Windows. In order to make the differentiation of these versions easier, all system dependent source files have got an OS Postfix (see 2.2.3). Via button **Delete** an entry can be removed from the list. The order of the path entries within the list is without any effect but can be modified by selecting an entry and using the buttons **Move up** resp. **Move down**.

In the description of a component a placeholder is assigned to the system-dependent source files. The OS Postfix will replace this placeholder and thus the complete name of the file will get obvious.

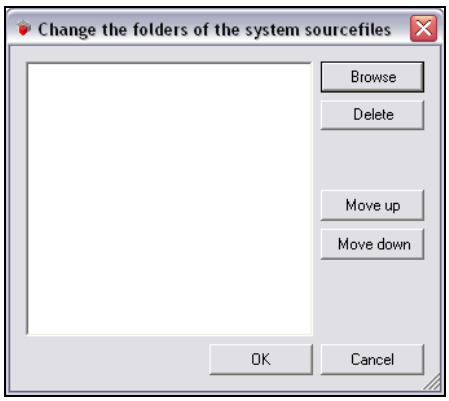


Fig.16 Change the folders of the system source files

A new path can be chosen in a dialog which is to be opened via button **Browse**. After having closed this dialog with OK, the path will be added to the list and can be edited there.

Via button **Delete** an entry can be removed from the list. The order of the path entries within the list is without any effect but it can be modified by selecting an entry and using the buttons **Move up** resp. **Move down**.

4.2.3 'Project options'

The dialog contains the following project-dependent options:

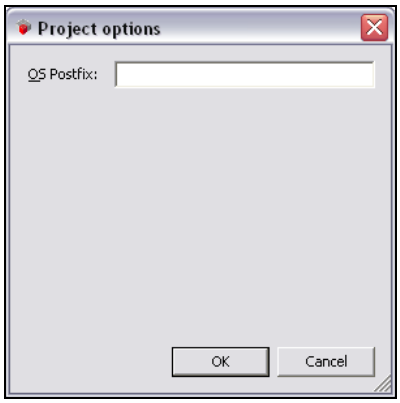


Fig.17 Project Options

OS Postfix: The OS Postfix serves to find the system-dependent source files which are needed for a specified system. The Postfix replaces the placeholder in system-dependent source files and thus a file name will be created as a composition from the entry for the system-dependent source file and the postfix. Example for OS Postfix: “Win32”.

4.3 Output

4.3.1 'All output files'

This command (menu ‘Output’) will create – if the configuration is without detected errors – an output file for all active backend-implementations. Such output files will be stored in the same folder as the project. The file name is composed of the projectname and an extension which is specific for the respective backend-implementation. For example for backend “ComponentList” a file „<projectname.cfg>“ will be created.

Command 'All output files' creates the following files:

- 'C – filenames'
- 'C – sources'
- 'Component init functions'
- 'Component list'
- 'Documentation'
- 'Interface not implemented list'
- 'MSVC source files'
- 'Windows makefile'

4.3.2 'C – filenames'

File extension: "._c"

The file names of all source files, which are used by the active components, will be listed.

4.3.3 'C – sources'

File extension: ".lst"

All source files which are used by the active components will be listed alphabetically. The path of the source files will be dumped relatively to the location of the output file. The output has the following structure:

```
SRCS =    ../../components/cmpaddrsrvc/cmpaddrsrvc.c \
          …… \
          ../../components/cmpapp/cmpapp.c
```

4.3.4 'Component init functions'

File extension: ".h"

For each active component the initialization is created, composed of two parts: 1.the declaration and 2. the call of the initializing function. In the following see an example for the content of the file:

```
#define COMPO_INIT_DECLS \
int SysWindowFileDialog__Entry(INIT_STRUCT *pInitStruct); \
… \
int SysWindow__Entry(INIT_STRUCT *pInitStruct); \
```

```
#define COMPO_INIT_LIST \
{"SysWindowFileDialog", SysWindowFileDialog__Entry, 0}, \
… \
{"", NULL, 0}
```

4.3.5 'Component list'

File extension: ".cfg"

All active components are dumped in a numbered list, sorted alphabetically according to the following format:

```
Component.1=SysWindowFileDialog
…
Component.30=CmpVisuServer
```

This list e.g. can be used in the configuration file of the runtime system.

4.3.6 'Documentation'

File extension: ".pdf" und ".xml"

A **documentation file** in PDF format will be created, containing the documentation of the active components and of the interfaces which are implemented in the components.

4.3.7 'Interface not implemented list'

File extension: "..._NotImpl.h"

For each interface which needs one of the active components it will be checked whether an other active component also implements this interface. All interfaces which are implemented by no active component, will be marked as “not implemented” in this file; e.g. by the following entry:

```
#define CMPSCHEDULE_NOTIMPLEMENTED
```

4.3.8 'MSVC source files'

File extension: ".ds_"

All used source files will be specified and stored in a notation which can be imported in a Visual C++ 6 project. Relative paths are used for this purpose.

```
# Begin Source File

SOURCE=..\..\components\systimer\systimer.c
# End Source File
```

4.3.9 'Windows makefile'

File extension: ".mk"

A part of a Windows Make File will be created, with relative paths in Windows notation. The output of the dependencies looks like follows:

```
CmpGateway.o : ..\..\components\cmpgateway\cmpgateway.c
```

4.3.10 'Copy C – sourcefiles'

This function copies the used C – source files without the previous folder structure to a folder which can be specified. Already existing versions of a source file in the specified folder will be overwritten. If no path is configured in the project settings, a folder with name "projectname_Sources" will be created parallel to the location of the rcp-file, and the source files will be stored there.

4.4 General Informationen

All commands referring to general information are available in the ‚?' menu.

4.4.1 ‚?' – ‚Help'

Command Help in menu '?' or using key <F1> opens the online help in the HTML Help Viewer (as from Internet Explorer V4.1).

The **Content** tab provides the contents tree of the online help. The particular **books** can be opened resp. closed via the preceded plus- and minus signs. The **page** currently selected in the content tree will be displayed in the right part of the help view.

Links within the text, leading to other help pages, resp. expandable text parts or images are indicated by a different color and underlining. A mouse-click on such text strings opens the concerned page resp. expands the paragraph.

In the **Index** tab you can search the help for a certain keyword, in the **Searc**h tab a full-text search can be initiated. Follow the instructions given on the tabs.

The Glossary tab provides descriptions for some terms.

4.4.2 ‚?' – ‚About'

This command opens a window providing information on the RtsConfigurator.

5 Errors and Warnings

5.1 Note concerning errors and warnings

As long as errors are detected or not all required functions are implemented by the active components, no output files can be created. Errors and warnings are displayed in the message window.

The message window and both lists of missing functions will be updated by one of the following actions:

- Opening a project
- Activating / deactivating of a component (even if this is done via the function bar)
- Changing the component paths
- Changing the paths for system-dependent source files
- Changing a source file in a component (only update of the message window)

5.2 Warnings

5.2.1 Warning: The access to the following folder was denied

You have specified a path in the list of ‚Component folders' or in the list of ‚Folder of system components', which you can not be access due to the permissions of your user account. In the case of the 'Component folders' also a sub-folder of the given path can be concerned.

5.2.2 Warning: The following folder was not found

In the ‚Component folders' list or in the ‚Folder of system components' list a path is specified which is not available on your system.

5.2.3 Warning: The interface file <interfacefilename> used in the component <componentname> can not be found.

The interface file specified in the component could not be found. The file name of the interface file is defined in the dependency files. This is only a warning, because the component is not active.

5.3 Errors

5.3.1 Error: The function <functionname> is required by …

The function is required by one or several components, but no component implementing the function is active. The components are displayed in the ‚Ambiguous dependencies' list and can be selected and activated there.

5.3.2 Error: The component <componentname> is required by …

One or several components need the functions of the requested component. In the ‚Definite dependencies' list the component can be selected and activated.

5.3.3 The function <functionname> … was not found

The required function cannot be found. The component requesting the function is not operative and should be deactivated, or a component implementing the function should be added to the configuration. To add a component to the configuration, the component path must be added to the ‚Components folder' list.

5.3.4 Error: No OS Postfix is available …

The OS Postfix is needed to find system-dependent source files. If it is not available those files will not be found. The OS Postfix can be specified via 'Options' → 'Project options'.

5.3.5 Error: The function <functionname> is implemented by more than one component, the components are …

The function is implemented by more than one component. To fix this error, only one of the listed components should be activated. The components can be deactivated in the ‚Available components' list.

5.3.6 Error: The sourcefile <sourcefilename> of the component <componentname> can not be found

The source file needed for the component could not be found. The file name is shown in the dependency file. You can fix the error by one of the following actions:

- Use 'Options' → 'Component folders…' resp. 'Folder of system components…' and enter there the path of the source file
- Expand the respective component, perform a double-click on the file name and select the source file manually

5.3.7 Error: The interface file <interfacefilename> used in the component <componentname> can not be found.

The interface file specified in the component could not be found. The file name of the interface file is specified in the dependency file. This is dumped as an error, because an active component is concerned.

5.3.8 Error: There are no components in the current project. It's not possible to generate output files.

Before output files can be created, there must be any components entered in the "Available components" list. For how to insert components, see chap. 4.2.1 'Component folders….

5.3.9 Error: The name of the project is invalid. It's not possible to generate output files.

Before output files can be created, the project at least once must have been stored with a correct project name. This is necessary, because the names of the output files are derived from the project name and the output files partially get stored in the same folder as the project. See also chap. 4.1.4 'Save As….

5.3.10 Error: The configuration of the runtimesystem is not correct. It's not possible to generate output files.

In the message window (see chap. 3.7 Message window) there are still error messages. All errors must be fixed before output files can be created.

5.3.11 Replace folders – Dialog

This dialog appears, when you load a project specifying paths which are not available on the system. This might occur if you copy the project to another system. In the **Replace folders** dialog such wrong paths can be replaced by appropriate new ones. After confirmation of the new settings the paths in the project will be replaced by the new ones and the project will be loaded..

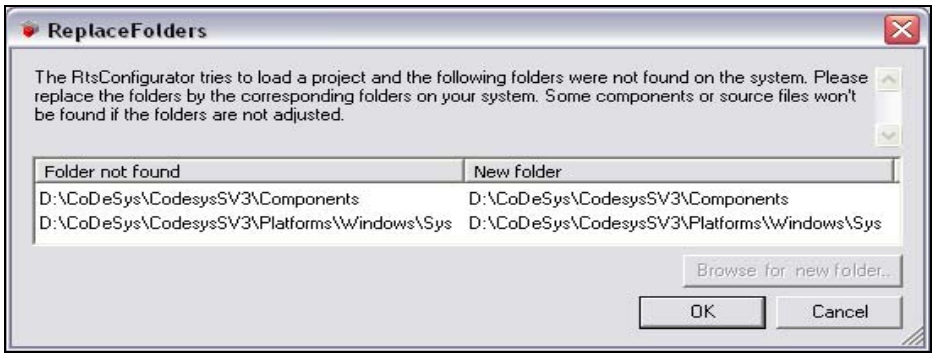


Fig.18 Replace folders

6 Glossary

Backend Implementation	Converts the data which are contained in the configuration project into an output file. The output file can have different purposes and thus have different contents.
Backend Interface	Interface to be provided by each backend implementation. Contains three methods: GetName, GetDescription, CreateOutput.
Dependency File	The dependency file contains the name of the component, the interface implemented by the component and all functions which are needed for this. The file name ends with ***Dep.m4.
Function	The runtime system calls functions which are provided by the components.
Interface File	In this file all interfaces provided by the respective component are described. The file name ends with ***Itf.m4.
Component	A component is a base module of the runtime system. It is described by its interface file and its dependency file.
Component list	The component list is the core part of the data management. In this list all components and their states are managed.
Project	In a project all important information is stored, so that by loading a project the previous configuration of a runtime system can be restored.

Index

Change History

Version	Description	Date
0.5	Translated according to preliminary version 0.4, which was reviewed and updated for use with CoDeSys SP V3.x in German (as from now only English version available)	09.01.2009
1.0	Formal Review and Release	20.01.2009