

RT-LAB

Version 10.5

User Guide

RTLAB-UG-105-00

I M A G I N A T I O N
T O
R E A L - T I M E



© 2007 Opal-RT Technologies Inc. All rights reserved for all countries.

Information in this document is subject to change without notice, and does not represent a commitment on the part of OPAL-RT Technologies. The software and associated files described in this document are furnished under a license agreement, and can only be used or copied in accordance with the terms of the agreement. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information and retrieval systems, for any purpose other than the purchaser's personal use, without express written permission of OPAL-RT Technologies Incorporated.

Documents and information relating to or associated with OPAL-RT products, business, or activities, including but not limited to financial information; data or statements; trade secrets; product research and development; existing and future product designs and performance specifications; marketing plans or techniques, client lists, computer programs, processes, and know-how that have been clearly identified and properly marked by OPAL-RT as "proprietary information," trade secrets, or company confidential information. The information must have been developed by OPAL-RT and is not made available to the public without the express consent of OPAL-RT or its legal counsel.

ARTEMIS, RT-EVENTS, RT-LAB and DINAMO are trademarks of Opal-RT Technologies, Inc. MATLAB, Simulink, Real-Time Workshop and SimPowerSystem are trademarks of The Mathworks, Inc. LabVIEW is a trademark of National Instruments, Inc. QNX is a trademark of QNX Software Systems Ltd. All other brand and product names are trademarks or service marks of their respective holders and are hereby acknowledged.

We have done our best to ensure that the material found in this publication is both useful and accurate. However, please be aware that errors may exist in this publication, and that neither the authors nor OPAL-RT Technologies make any guarantees concerning the accuracy of the information found here or in the use to which it may be put.

Published in Canada

Contact Us

For additional information you may contact the Customer Support team at Opal-RT at the following coordinates:

Tool-Free (US and Canada)	1-877-935-2323 (08:30-17:30 EST)
Phone	1-514-935-2323
Fax	1-514-935-4994
E-mail	support@opal-rt.com info@opal-rt.com sales@opal-rt.com
Mail	1751 Richardson Street Suite 2525 Montreal, Quebec H3K 1G6
Web	www.opal-rt.com

CHAPTER 1: INTRODUCTION

CHAPTER 2: GETTING STARTED

Basic Tutorial	17
The Workbench	19
Editors and Views	21
Editors	23
Views	25
A simple project	27
Using the File menu	29
Using the popup	31
Using the New button	35
Preparing and building model	37
Selecting the development node.	39
Editing the model	41
Preparing the build configuration	43
Building the model	45
Assigning subsystems	47
Executing model	49
Selecting the simulation mode	51
Loading the model	53
Running the model	55
Using the Console.	57
Resetting the model	59
Closing and editor	61
Navigating resources	63
Opening resources in the Project Explorer	65
Files	67
Deleting resources	69
Copying and renaming	71
Copying.	73
Renaming	75
Searching	77
Rearranging views and editors	79
Drop cursors	81
Rearranging views	83
Rearranging tabbed views	85
Tiling editors	87
Maximizing and minimizing elements of the workbench presentation	89
Perspectives	93
New perspectives	95
Saving perspectives	97
Configuring perspectives	99

Comparing	101
Simple compare	103
Understanding the comparison	105
Working with comparison	107
Local history	109
Responsive UI	111
Exiting the Workbench	113
Welcome	115

CHAPTER 3: CONCEPTS

Workbench	119
Resources	121
Resource hierarchies	123
Linked resources	125
Working Sets	127
Local history	129
Perspectives	131
Editors	133
External editors	135
Model Editor	137
Model Overview Page	137
Development Page	139
. Execution Page	142
Environment Variables Page	144
Files Page	147
Assignation Page	149
Diagnostic Page	152
Hardware Page	153
Simulation Tools Page	154
Target Editor	157
Target Overview Page	157
Diagnostic Page	159
Simulation Settings Page	160
Software Page	160
Views	163
Fast views	165
Detached views	167
Project Explorer	169
Compilation View	189
Display View	193
Variable Viewer	199
Properties View	203
Matlab View	211

Console View	215
Interactive Python Console	219
Target Console	223
Log Console	225
Variables Table View	227
Probe Control Panel	235
Monitoring View	241
Terminal View	247
Help view	253
Search View	257
Workbench Menus	277
File menu	279
Edit menu	281
Search menu	283
Navigate menu	285
Tools menu	287
Simulation menu	289
Window menu	293
Help menu	297
Toolbars	299
Main Toolbar	301
Wizards	303
New RT-LAB project wizard	305
New RT-LAB model wizard	309
Import wizard	311
Add Model wizard	313
New Target wizard	317
Detected Targets Wizard	321
New Resource Project wizard	325
New Folder wizard	327
New File wizard	329
Existing RT-LAB model import wizard	331
Export wizard	335
Flash bitstream wizard	337
Connect to Embedded Simulation Wizard	341
Load Parameters wizard	345
Save Parameters wizard	351
Dialogs	353
Build Configurations	355
File Search	357
RT-LAB Search	359
License Request Dialog	363
Preferences	365
RT-LAB Preferences Pages	367

Capabilities preference page	369
Compiler and Linker preference page	371
Monitoring preference page	373
Debugging preference page	375
Environment Variables preference page.	377
Files transfers preference page	379
Performance preference page.	381
Real-time preference page.	383
Hardware preference page.	385
Simulation Tools preference page.	387
EMTP preference page.	389
Matlab preference page.	391
RT-LAB preference page	393
Target Detection preference page.	395
General preferences pages	397
Accessibility preference page	399
Annotations preference page	401
Capabilities preference page	403
Appearance preference page	405
Colors and Fonts preference page.	407
Compare/Patch preference page.	409
Content Types preference page	411
Editors preference page	413
File Associations preference page	415
Help preferences	417
General preference page	419
Help Content preference page	421
Keys preference page	423
Label Decorations preference page.	427
Linked Resources preference page	429
Local History preference page	431
Perspectives preference page.	433
Quick Diff preference page	435
Search preference page.	437
Spelling preference page	439
Startup and Shutdown preference page.	441
Text Editors preference page	443
Web Browser preference page	445
Workspace preference page.	447
Cheat Sheets	449
Launching a cheat sheet.	449
Starting the cheat sheet	449
Restarting the cheat sheet	450
Progressing through the steps.	450

Getting help information for tasks	450
Skipping a step	450
Redoing a step	450
Closing the cheat sheet	450
MetaController	451
Help	453

CHAPTER 4: TASKS

Building models	457
Building a simple model for RT-LAB	457
Building a distributed Model for RT-LAB	458
Executing models	465
Target platform	465
Simulation mode	465
Communication type	467
Executing steps	468
Additional files	469
User script files	469
Debugging	470
Acquiring and Viewing Data	475
Acquisition Groups	475
Understanding Data Reception	476
Monitoring Models	481
Monitoring Overview	481
Monitoring Architecture and Concepts	481
Enabling Monitoring	482
Using the Monitoring View	482
Using the OpMonitor Block	484
Measuring a Model's Subsystem Calculation Time	484
Measuring User Code Source Calculation Time	484
Events and Probes List	485
Using Python Script and the Macro Recorder	489
Using RT-LAB Blocks	495
Generic blocks	495
I/O blocks	495
RT-LAB Connectivity	497
RT-LAB Orchestra	499
Introduction	499
Configuring the RT-LAB Orchestra communication layer	499
DDF Configurator	503
RT-LAB Orchestra Simulink blocks	506
Setting up an RT-LAB Orchestra co-simulation	507
Running an RT-LAB Orchestra co-simulation	508

RT-LAB Asynchronous processes	509
Introduction	509
Architecture overview	509
Building the asynchronous application.	510
Running the Asynchronous program	515
RT-LAB User SFunction.	517
Introduction	517
Changes Resulting from the Replacement of SimStruct with the rtModel517	
RT-LAB / Xilinx System Generator toolbox integration	519
Using ScopeView	523
Embedding Simulation	529
Taking a Snapshot	533
Registering a User S-Function for Snapshot	533
Working with perspectives.	535
Switching between perspectives	537
Specifying the default perspective	539
Opening perspectives.	541
Changing where perspectives open	543
Configuring perspectives	545
Saving a user defined perspective	547
Deleting a user defined perspective	549
Resetting perspectives	551
Working with views and editors	553
Opening views.	555
Moving and docking views	557
Rearranging tabbed views.	559
Creating fast views	561
Working with fast views	563
Detaching views	565
Opening files for editing	567
Editing files outside the Workbench	569
Tiling editors	571
Maximizing and minimizing elements of the workbench presentation	573
Customizing the Workbench	577
Customizing the Welcome.	579
Workspace Switching	581
Rearranging the main toolbar	583
Changing the key bindings	585
Changing font and colors	587
Changing the placement of the tabs.	589

CHAPTER 5: LEGAL

Introduction

1.1 About RT-LAB

RT-LAB™ is a distributed real-time platform that facilitates the design process for engineering systems by taking engineers from Simulink dynamic models to real-time with hardware-in-the-loop, in a very short time, at a low cost. Its scalability allows the developer to add compute-power where and when needed. It is flexible enough to be applied to the most complex simulation and control problem, whether it is for real-time hardware-in-the-loop applications or for speeding up model execution, control and test.

RT-LAB provides tools for running simulations of highly complex models on a network of distributed run-time targets, communicating via ultra low-latency technologies, in order to achieve the required performance. In addition, RT-LAB's modular design enables the delivery of economical systems by supplying only the modules needed for the application in order to minimize computational requirements and meet customers price targets. This is essential for high-volume embedded applications.

1.2 Key Features

Fully integrated with MATLAB/Simulink

All model preparation for RT-LAB is done with established dynamic system modeling environments, which allows the user to leverage experience in using these tools.

Specialized blockset for distributed processing, inter-node communication and signal I/O

RT-LAB provides tools for easy separation of the system model into subsystem models that can be executed on parallel target processors (standard PCs running either the QNX Real-Time operating system, or Red Hat Linux). In this way, if you need to run a model in real-time that cannot be run on a single processor, RT-LAB provides a means of sharing the load over several processors.

Fully integrated with third-party modeling environments and user code libraries

RT-LAB supports models from StateFlow, StateMate, CarSim RT, GT-Power RT, AMESim, Dymola, as well as legacy code in C, C++ and FORTRAN.

Comprehensive API for developing your own on-line application.

Using environments such as LabVIEW, C, C++, Visual Basic, TestStand, Python and 3D virtual reality tools it is possible to create custom user and test automation interfaces.

Off-the-shelf technologies

RT-LAB is the first fully scalable simulation and control package that allows you to separate models for execution in parallel on a network of standard desktop PCs, PC/104s or on SMP (symmetric multi-processor) servers.

Driven by the demands of a mass market, users take advantage of rapid advancements in a wide range of readily available technologies, as well as relatively low costs. RT-LAB uses standard Ethernet and FireWire (IEEE 1394) communications, and an extensive range of ISA, PCI, PXI and PCMCIA analog and digital I/O boards.

Shared Memory, FireWire, InfiniBand or UDP/IP interprocessor communication

At execution time, RT-LAB provides seamless support for inter-processor communication, using any combination of UDP/IP, Shared Memory and all readily available technologies for low-latency communication of data between the target processors. You can also interact with the simulation in real-time from the host station using TCP/IP.

Integrated interface for signal and parameter visualization and control

With RT-LAB's visualization and control panel, you can dynamically select signals to trace, modify any model signal or parameter in real-time.

Extensive I/O card support - over 100 devices supported

RT-LAB integrates with Opal-RT's OP5000 hardware interface devices for nanosecond precision timing and real-time performance. RT-LAB also supports cards from other leading manufacturers such as National Instruments, Acromag, Softing and SBS.

Choice of RTOS: QNX, Red Hat Linux or Windows (for software real-time)

RT-LAB is the only real-time simulation framework that offers you a choice of two high-performance Real-Time Operating Systems (RTOS). RT-LAB is available for QNX an RTOS with a proven track record for mission-critical engineering applications - and Red Hat Linux - the popular open-source Linux operating system. RT-LAB also offers Windows as a soft-real-time RTOS.

Optimized Hard-Real-Time Scheduler – high performance, low jitter

Within a time step, the system is doing more than computing the dynamic model. It also does administrative tasks, such as reading and writing I/O, updating the system clock, scheduling tasks, logging data, and handling communications. This restricts the amount of time available within a frame to compute the model values limiting the size of model that can be computed on a single processor. RT-LAB has reduced this overhead to a few percent of raw hardware performance without losing functionality, thereby increasing the capacity to compute more complex models.

High speed XHP Mode – Multirate XHP Mode – Software-Synchronized mode

RT-LAB XHP (eXtra High Performance) mode allows very fast computation of the real-time model on the target system. This has allowed our customers to simulate complex systems over distributed processors, with analog and digital I/O, at cycle times below 10 microseconds.

The XHP Mode of RT-LAB can slash scheduling overhead to less than a microsecond, letting you use the full power of your system for the computation of highly dynamic models in real-time which is a solution for the developers who have the constant challenge of achieving accurate, high fidelity responses for the real-time simulation increasingly complex models. Even when the signals in a hardware-in-the-loop (HIL) system need only be updated in the 100s of microseconds time frame, the model may need to be computed many times between each major time step in order to maintain numerical accuracy. The XHP Mode, by far, out-performs any other real-time system, and is particularly useful for modeling electrical systems, such as drive controls and power electronics.

1.3 Intended Audience and Required Skills and Knowledge

1.3.1 MATLAB

MATLAB is a technical computing software package that integrates programming, calculation and visualization. MATLAB also includes Simulink; this software package is discussed below. As RT-LAB works in conjunction with these environments to define models, you must be familiar with aspects of MATLAB as related to Simulink.

If you are intending on gathering data from one system and then process the data offline, for example, you must know how to save the data and retrieve it using MATLAB and to display data through the different on-screen tools available within the software.

1.3.2 Simulink

Simulink is a software package that enables modeling, simulation and analysis of dynamic systems. You describe its models graphically, following a precise format based on a library of blocks. RT-LAB uses Simulink to define models that will be executed by the real-time multiprocessing system and defines its own simulation parameters through Simulink's. It is expected that you have a clear understanding of Simulink's operation, particularly regarding model definition and the model's various simulation parameters. If you intend to create your own blocks to interact with RT-LAB, you should know how to create Simulink icons (S-functions) for both the command station and target environments.

1.4 Organization of this Guide

There are several guides offered in the list of RT-LAB documentation:

- Installation Guide
- API reference Guide
- User Guide

This document is the user guide. The topics covered are:

- **Introduction on page 9**- Provides an introduction to simulation and the principles behind RT-LAB.
- **Getting Started on page 15** - Presents tutorials on the RT-LAB workbench and RT-LAB simulator.
- **Concepts on page 117** - Provides a complete description of all concepts of RT-LAB, as views or editors, related to the workbench and the simulator.
- **Tasks on page 457** - Describes how to realize tasks, as building or executing a model, using the RT-LAB workbench.

1.5 Conventions

Opal-RT guides use the following conventions:

Table 1: General and Typographical Conventions

THIS CONVENTION	INDICATES
Bold	User interface elements, text that must be typed exactly as shown.
Note:	Emphasizes or supplements parts of the text. You can disregard the information in a note and still complete a task.
Warning:	Describes an action that must be avoided or followed to obtain desired results.
Recommendation:	Describes an action that you may or may not follow and still complete a task.
Code	Sample code.
<i>Italics</i>	Reference work titles.
Blue Text	Cross-references (internal or external) or hypertext links.

1.6 Basic concepts

This section describes the basics of RT-LAB, provides an overview of the simulation process, from designing and validating the model, to using block diagrams and I/O devices, to running the simulation and using the Console as a graphic interface.

1.6.1 Designing and Validating Models

The starting point for any simulation is a mathematical model of the system components that are to be simulated.

You design and validate a model by analyzing the system to be modelled and implementing the model in the dynamic simulation software. RT-LAB is designed to automate the execution of simulations for models made with offline dynamic simulation software, like Simulink, in a real-time multiprocessing environment.

RT-LAB is fully scalable, enabling you to separate mathematical models into blocks to be run in parallel on a cluster of machines, without changing the model's behavior, introducing real-time glitches, or causing deadlocks.

1.6.2 Using Block Diagrams

Using block diagrams for programming simplifies the entry of parameters and guarantees complete and exact documentation of the system being modeled.

Once the model is validated, you separates it into subsystems and insert appropriate communication blocks. Each subsystem is executed by target nodes in RT-LAB's distributed system.

1.6.3 Using I/O Devices

RT-LAB supports the use of Input/Output devices to enable the integration of external physical components into the system. This arrangement is commonly known as a **Hardware-in-the-Loop (HIL)** configuration or rapid control prototyping (RCP) whether the plant or controller is simulated, respectively.

The optimized use of I/O devices enables RT-LAB to work as a programmable control system that presents a flexible real-time user-machine interface.

Interfaces for I/O devices are configured through custom blocks that need only be added and connected to the graphic model's blocks. RT-LAB's automatic code generator will map the model's data onto the physical I/O cards.

1.6.4 Running Simulations

Once the original model is separated into subsystems associated with the various processors, each portion of the model is automatically coded in C and built for execution by the target nodes.

Target nodes are commercial PCs, equipped with PC-compatible processors, that operate under a Windows XP/Vista/7, QNX or Red Hat Linux environment.

1.6.5 Executing and Manipulating Model Parameters

When the C coding and compilation are complete, RT-LAB automatically distributes its computation among the target nodes and provides an interface so you can execute the simulation and manipulate the model's parameters. The result is high-performance simulation that can run in parallel and in real-time.

1.6.6 Using the Console as a Graphic Interface

You can interact with RT-LAB during a simulation by using the console, a command terminal operating on the command station. Communication between the console and the target nodes is performed through a TCP/IP connection. This enables you to save any signal from the model, for viewing or for

offline analysis. It is also possible to use the console to modify the model's parameters while the simulation is running.

1.6.7 Working with RT-LAB

RT-LAB software runs on a hardware configuration consisting of the following components:

- Command station
- Compilation node
- Target nodes
- I/O boards

RT-LAB software is configured on a Windows XP/Vista/7 or Red Hat linux computer called the **command station**. The Command Station is a PC workstation that serves as your interface. The Command Station enables you to:

- edit and modify models;
- see model data;
- run the original model under its simulation software (Simulink);
- distribute code;
- control the simulator's Go/Stop sequences.

Simulations can be run entirely on the command station computer, but they are typically run on one or more target nodes.

For real-time simulation, the preferred operating system for the target nodes is **QNX** or **Red Hat Linux**. When there are multiple target nodes, one of them is designated as the compilation node. The Command Station and target node(s) communicate with each other using communication links and for hardware-in-the-loop simulations target nodes may also communicate with other devices through I/O boards.

The target nodes are real-time processing and communication computers that use commercial processors.

These computers can include a real-time communication interface like **FireWire**, **Infiniband** or **cLAN** (depending on the selected OS), as well as I/O boards for accessing external equipment. The real-time target nodes perform:

- real-time execution of the model's simulation;
- real-time communication between the nodes and I/Os;
- initialization of the I/O systems;
- acquisition of the model's internal variables and external outputs through I/O modules;
- implementation of user-performed online parameters modification;
- recording data on local hard drive, if desired;
- supervision of execution of the model's simulation and communication with other nodes.

The compilation node is used to compile generated C code. Any target node could be the compilation node.

Various analog, digital and timer I/O boards are supported by RT-LAB. These enable connection to external equipment for applications such as HIL.

Getting Started

Contents:

- [Basic Tutorial](#)

Basic Tutorial

This tutorial provides a step by step walk-through of the RT-LAB Workbench.

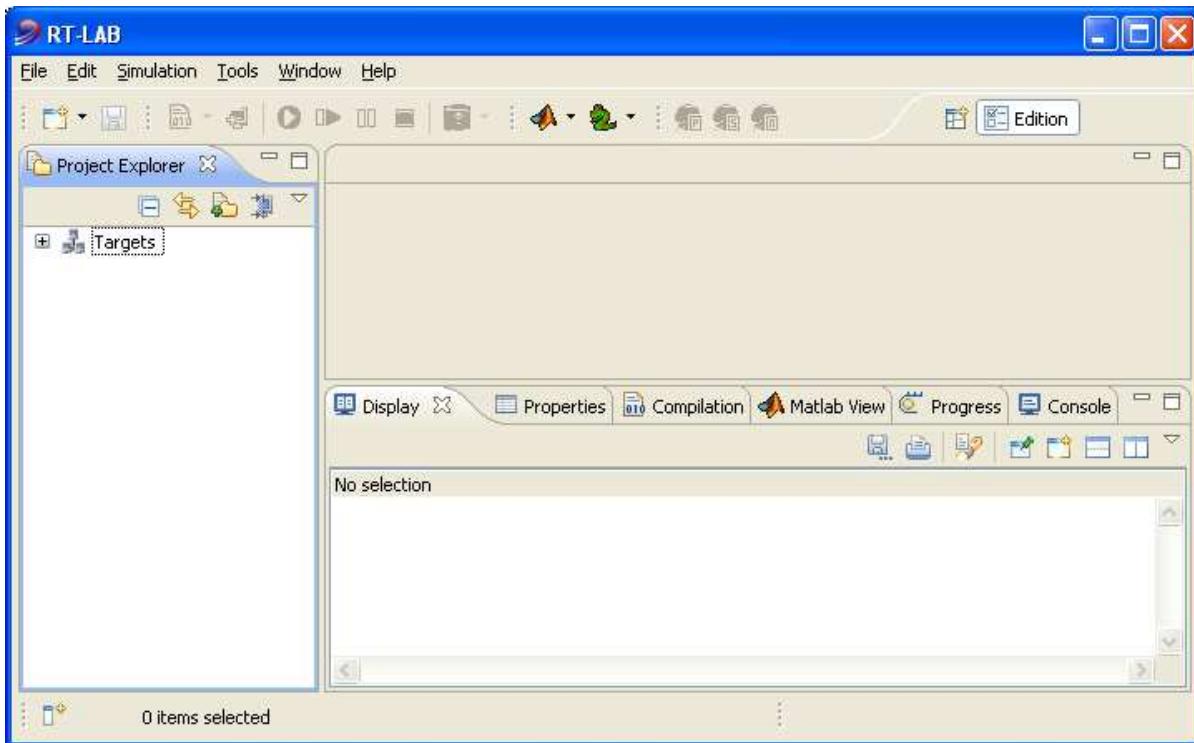
The Workbench

When RT-LAB is launched, the first thing you see is a dialog that allows you to select where the workspace should be located. The workspace is the directory where your work will be stored. For now, just click OK to pick the default location.

After the workspace location is chosen, a single Workbench window is displayed. A Workbench window offers one or more perspectives. A perspective contains editors and views, such as the Project Explorer. Initially, in the Workbench window that is opened, the RT-LAB Edition perspective is displayed, with only the Welcome view visible. Click the arrow labeled Workbench in the Welcome view to cause the other views in the perspective to become visible. Note you can get the Welcome view back at any time by selecting Help > Welcome.

A shortcut bar appears in the top right corner of the window. This allows you to open new perspectives and switch between ones already open. The name of the active perspective is shown in the title of the window and its item in the shortcut bar is highlighted.

You should be seeing the RT-LAB Edition perspective. The following views should be visible: The Project Explorer, Compilation, Display, Properties, Matlab, Progress, Console.



Editors and Views

Prior to commencing the RT-LAB Workbench tutorials found in this section, it is important to first be familiar with the various elements of the Workbench. A Workbench consists of:

- perspectives
- views
- editors

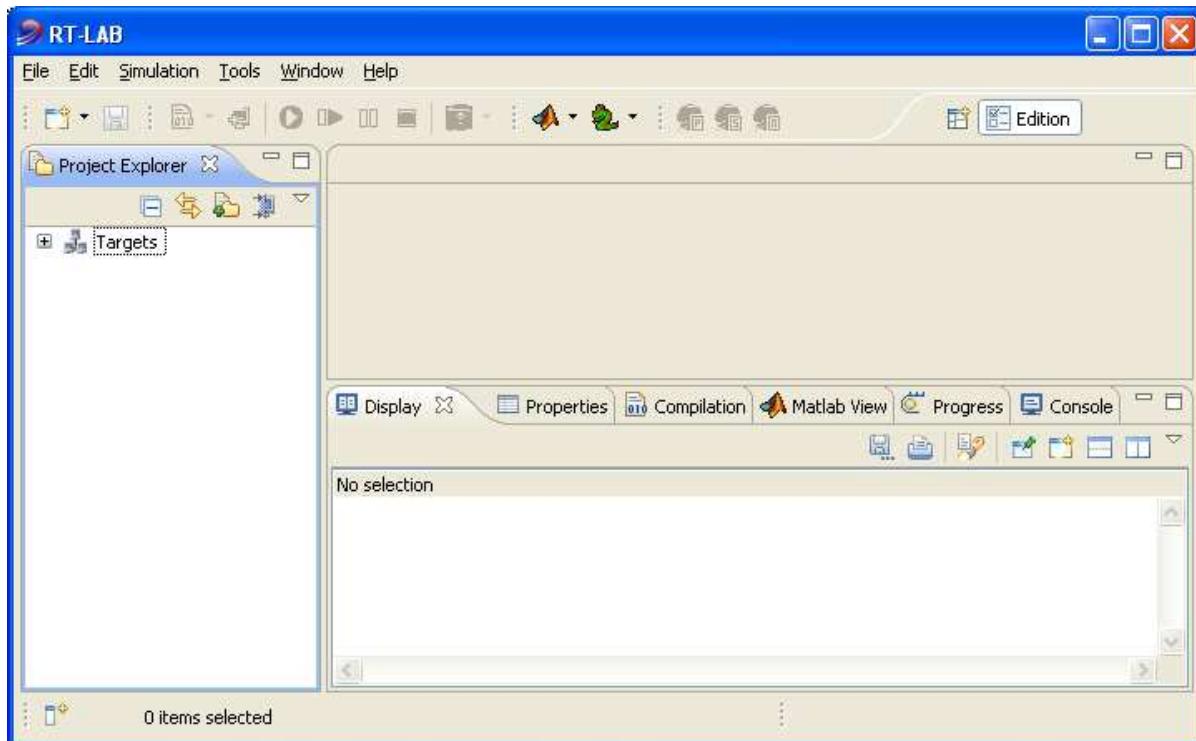
A perspective is a group of views and editors in the Workbench window. One or more perspectives can exist in a single Workbench window. Each perspective contains one or more views and editors. Within a window, each perspective may have a different set of views but all perspectives share the same set of editors.

A view is a visual component within the Workbench. It is typically used to navigate a list or hierarchy of information (such as the resources in the Workbench), or display properties for the active editor. Modifications made in a view are saved immediately.

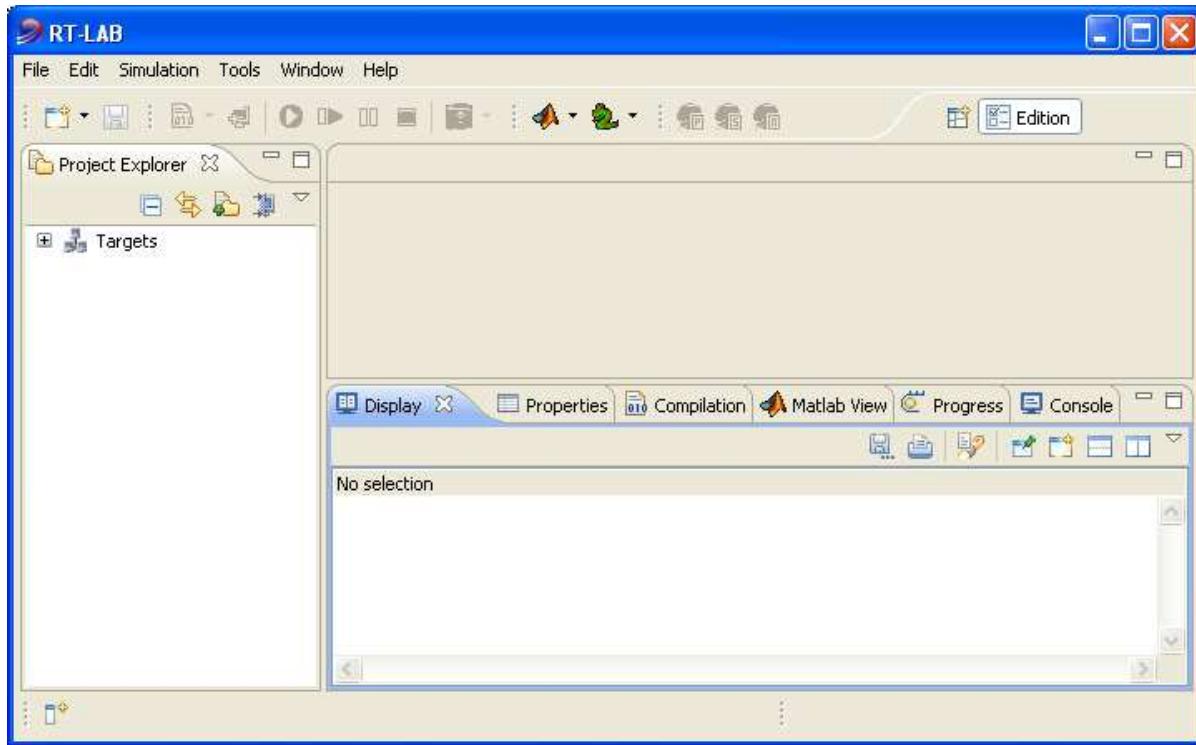
An editor is also a visual component within the Workbench. It is typically used to edit or browse a resource. The visual presentation might be text or a diagram. Typically, editors are launched by clicking on a resource in a view. Modifications made in an editor follow an open-save-close lifecycle model.

Some features are common to both views and editors. We use the term "part" to mean either a view or an editor. Parts can be active or inactive, but only one part can be active at any one time. The active part is the one whose title bar is highlighted. The active part is the target for common operations like cut, copy and paste. The active part also determines the contents of the status line. If an editor tab is not highlighted it indicates the editor is not active, however views may show information based on the last active editor.

In the image below, the Project Explorer view is active.

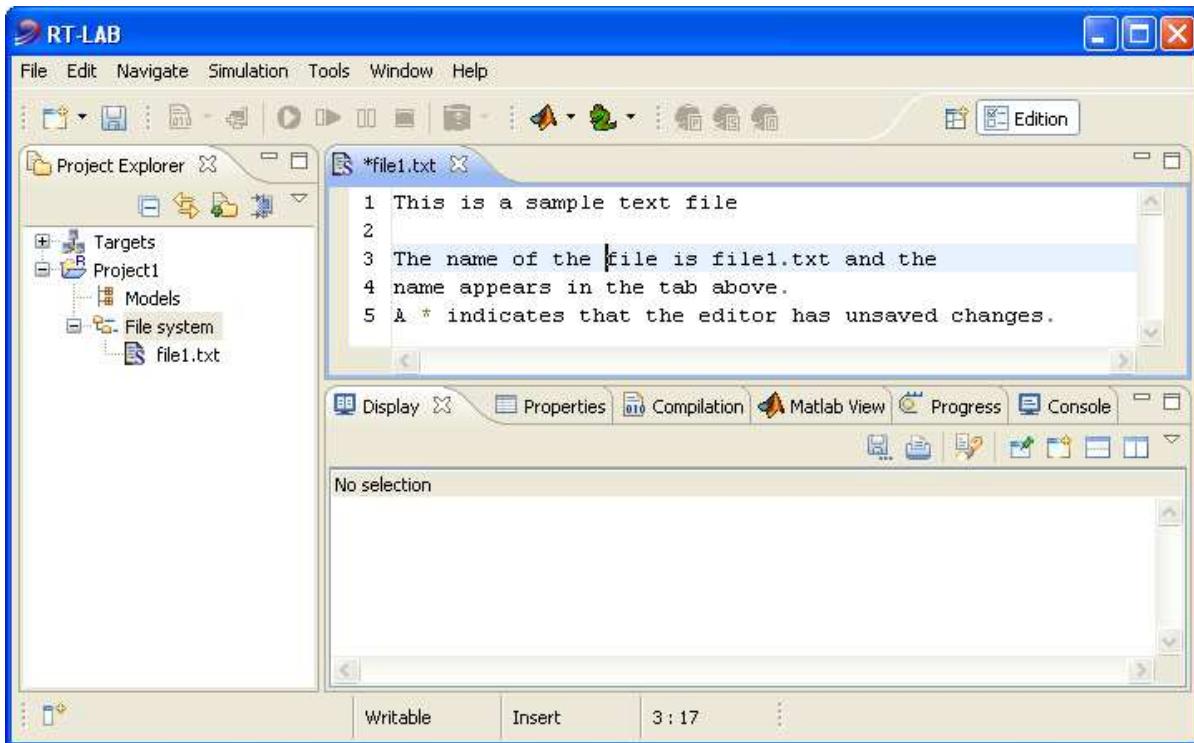


Clicking on the Display view causes the Display's title bar to become highlighted and the Project Explorer's title bar to no longer be highlighted, as shown below. The Display view is now active.



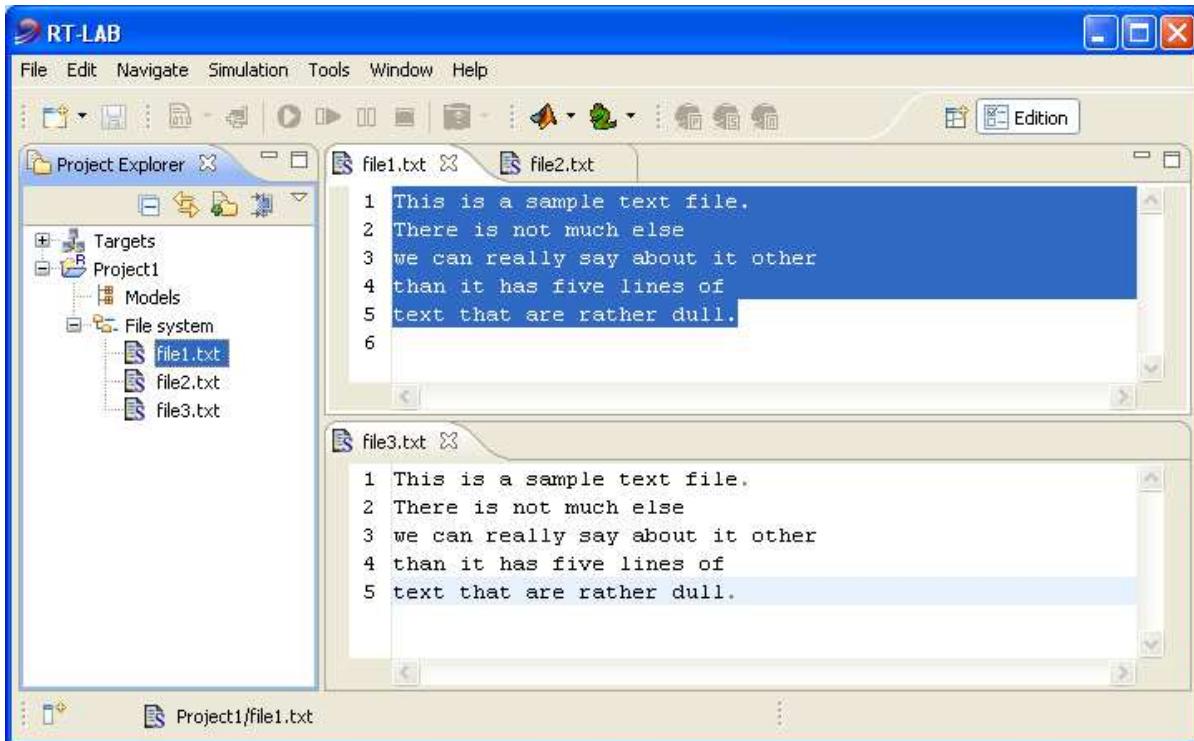
Editors

Depending on the type of file that is being edited, the appropriate editor is displayed in the editor area. For example, if a .TXT file is being edited, a text editor is displayed in the editor area. The figure below shows an editor open on the file file1.txt. The name of the file appears in the tab of the editor. An asterisk (*) appearing at the left side of the tab indicates that the editor has unsaved changes. If an attempt is made to close the editor or exit the Workbench with unsaved changes, a prompt to save the editor's changes will appear.



When an editor is active, the Workbench menu bar and toolbar contain operations applicable to the editor. When a view becomes active, the editor operations are disabled. However, certain operations may be appropriate in the context of a view and will remain enabled.

The editors can be stacked in the editor area and individual editors can be activated by clicking the tab for the editor. Editors can also be tiled side-by-side in the editor area so their content can be viewed simultaneously. In the figure below, editors for JanesFile.txt and JanesFile2.txt have been placed above the editor for JanesText.txt. Instructions will be given later in this tutorial explaining how to rearrange views and editors.



If a resource does not have an associated editor, the Workbench will attempt to launch an external editor registered with the platform. These external editors are not tightly integrated with the Workbench and are not embedded in the Workbench's editor area.

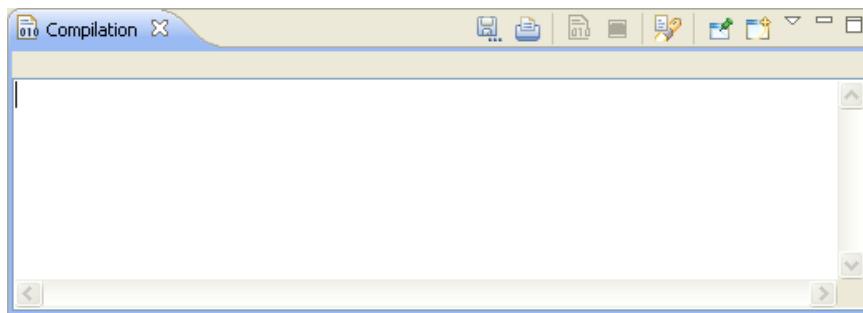
On Windows, if the associated editor is an external editor, the Workbench may attempt to launch the editor in-place as an OLE document editor. For example, editing a DOC file will cause Microsoft Word to be opened in-place within the Workbench if Microsoft Word is installed on the machine. If Microsoft Word has not been installed, Word Pad will open instead.

Views

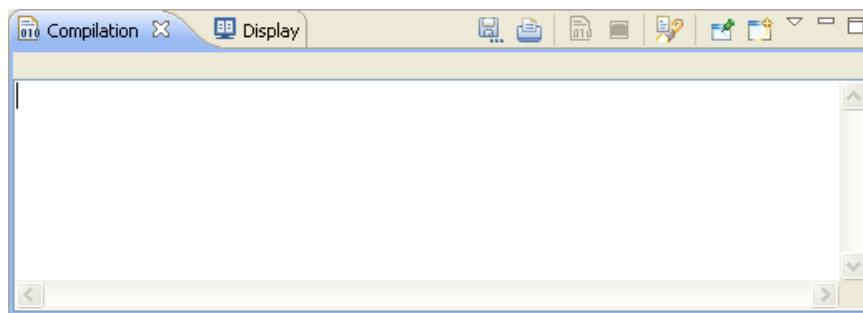
The primary use of Views is to provide navigation of the information in the Workbench. For example:

- The Project Explorer view displays the RT-LAB Workbench projects, their folders and files.
- The Compilation view displays the build outputs perform on models
- The properties view displays property names and values for a selected item such as a resource.

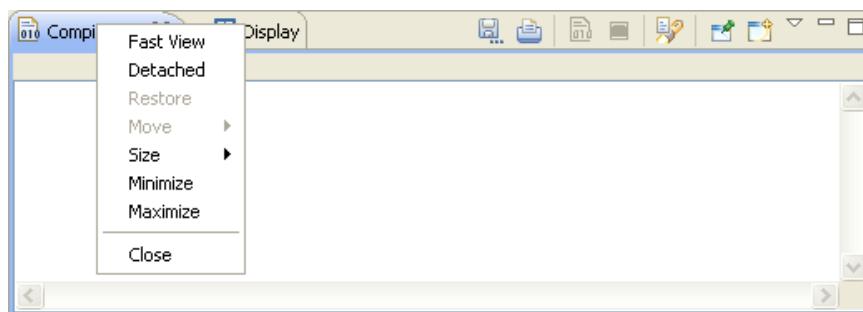
A view might appear by itself or stacked with other views in a tabbed notebook.



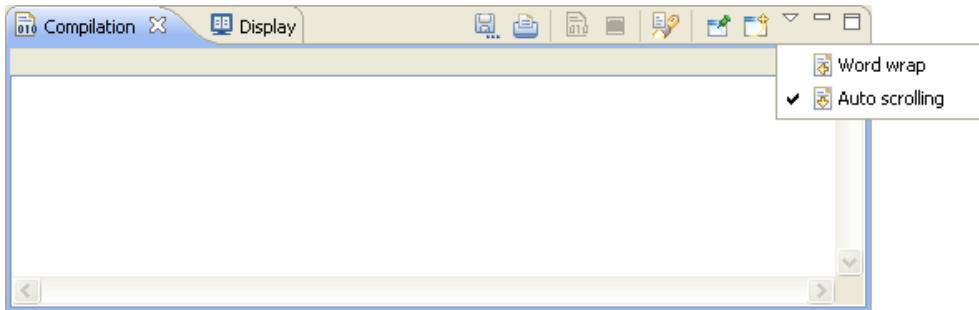
To activate a view that is part of a tabbed notebook simply click its tab.



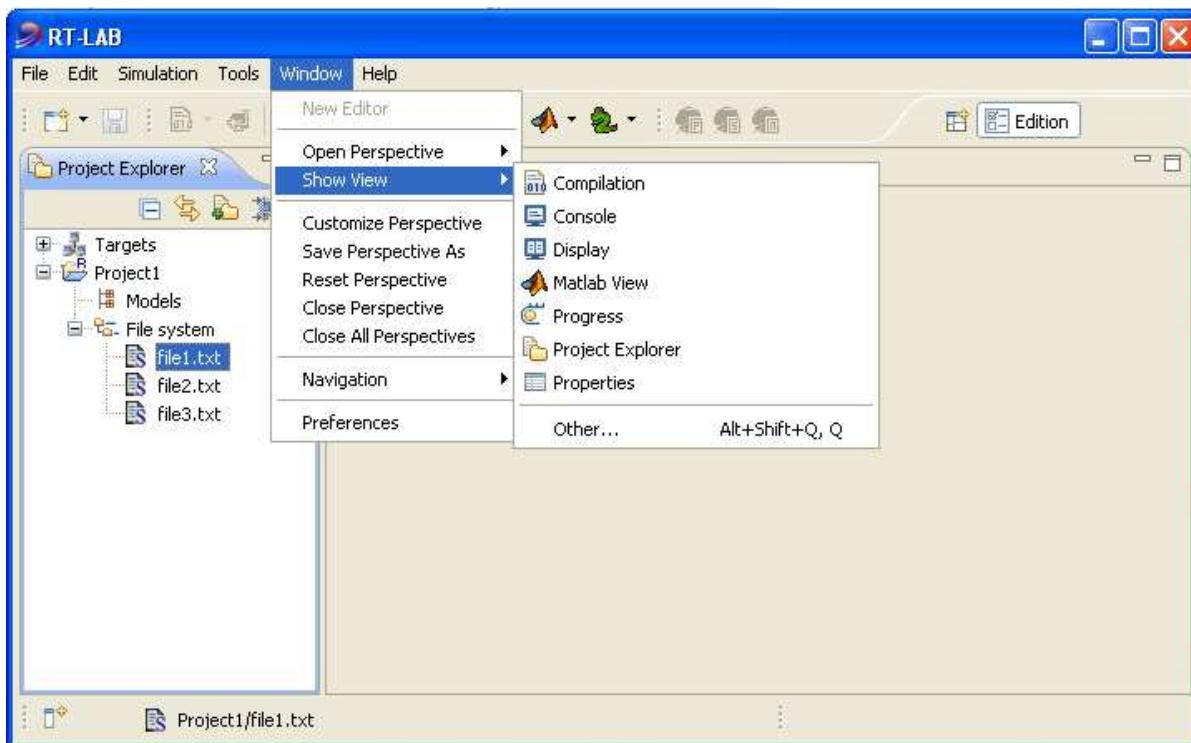
Views have two menus. The first, which is accessed by right clicking on the view's tab, allows the view to be manipulated in much the same manner as the menu associated with the Workbench window.



The second menu, called the "view pull-down menu", is accessed by clicking the down arrow . The view pull-down menu typically contains operations that apply to the entire contents of the view, but not to a specific item shown in the view. Operations for sorting and filtering are commonly found in the view pull-down.



A view can be displayed by selecting it from the **Window > Show View** menu. A perspective determines which views may be required and displays these on the **Show View** sub-menu. Additional views are available by choosing **Other...** at the bottom of the **Show View** sub-menu. This is just one of the many features that provide for the creation of a custom work environment.



Through the normal course of using the Workbench you will open, move, resize, and close views. If you'd like to restore the perspective back to its original state, you can select the **Window > Reset Perspective** menu operation.

A simple project

Now that the basic elements of the Workbench have been explained, here are some instructions for creating a simple project. New projects, folders, files, models and targets can be created using several different approaches. In this section resources will be created using three different approaches:

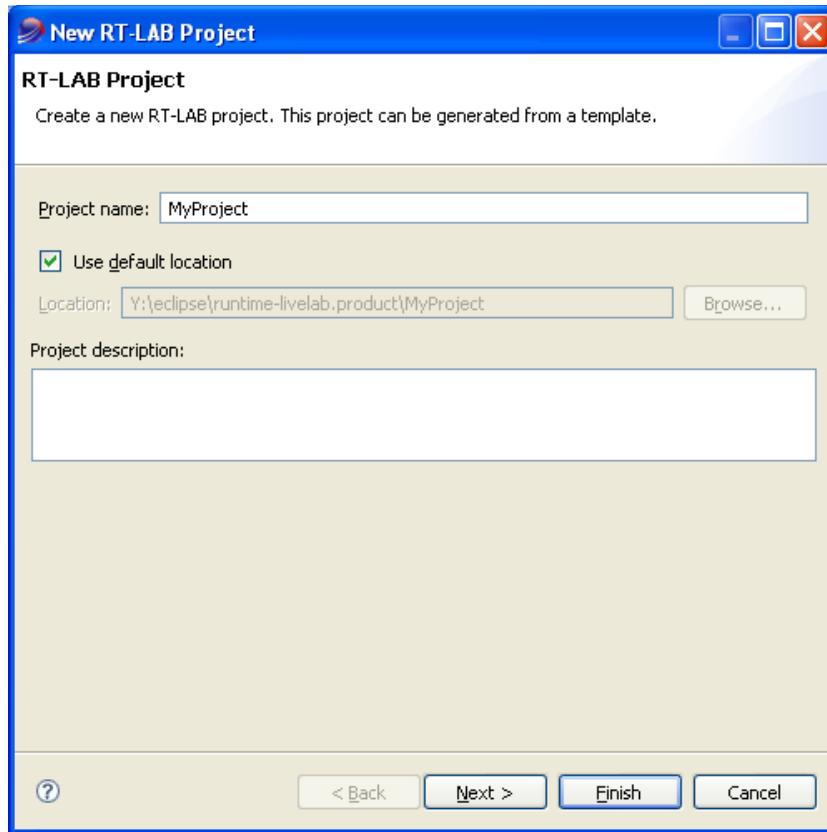
- **File** menu
- Project Explorer's view context menu
- New Wizard button

A project can be created using the **File** menu. Once the project has been created a model and target can be created as well.

Using the File menu

You can create new resources by using the **File > New** menu on the Workbench menu bar. Start by creating a simple project as follows:

- From the menu bar, select **File > New > RT-LAB Project...**
- In the **Project name** field, type your name as the name of your new project. Do not use spaces or special characters in the project name (for example, "MyProject").



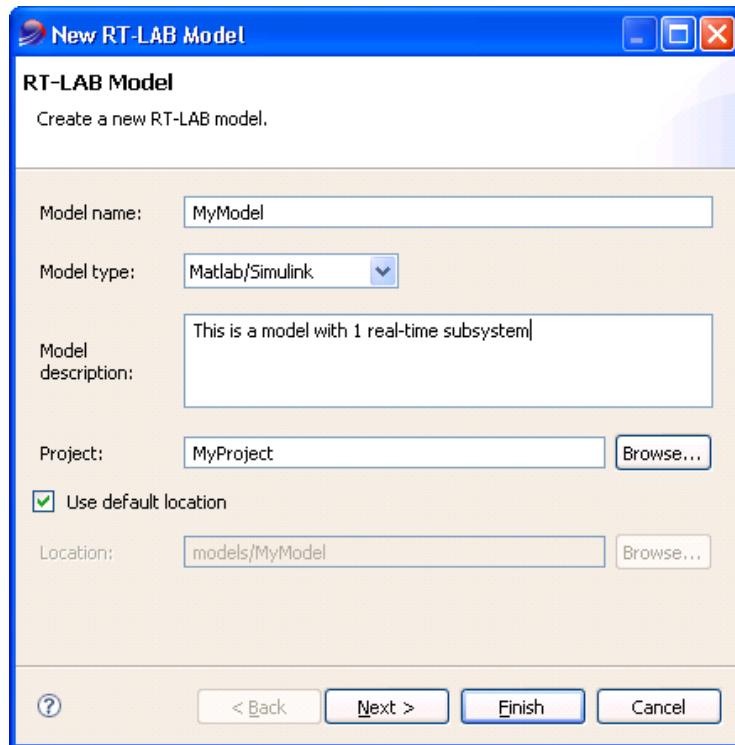
- Leave the box checked to use the default location for your new project. Click **Finish** when you are done.
- If you sneak a peek at the navigation view, you will see that it now contains the simple project we just created.



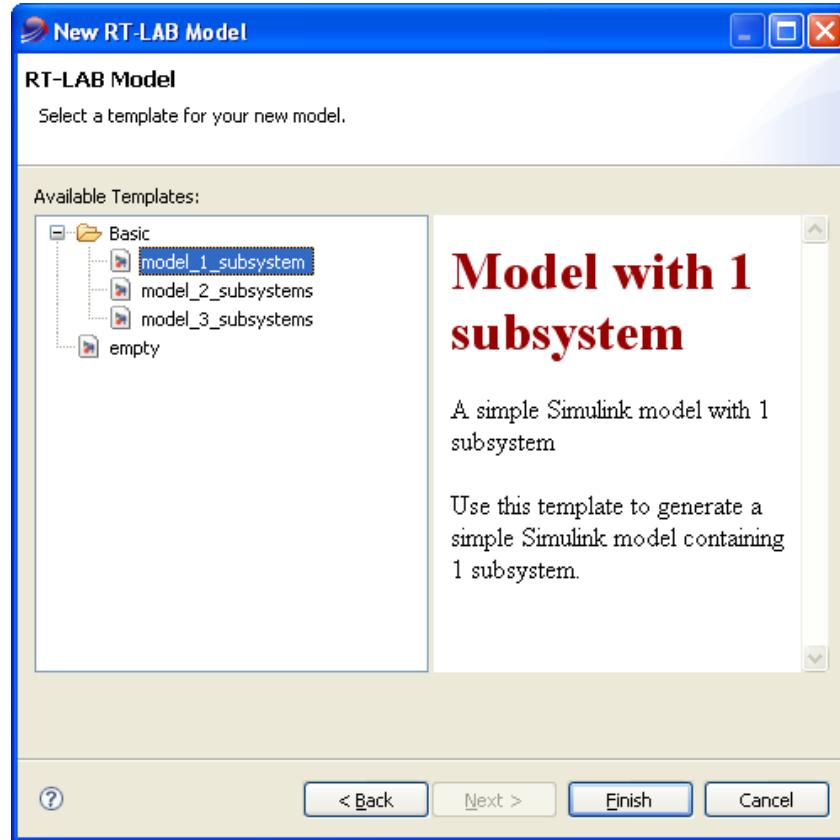
Using the popup

Now that we have our project we will create a model. We will create our model using the Project Explorer view's popup menu.

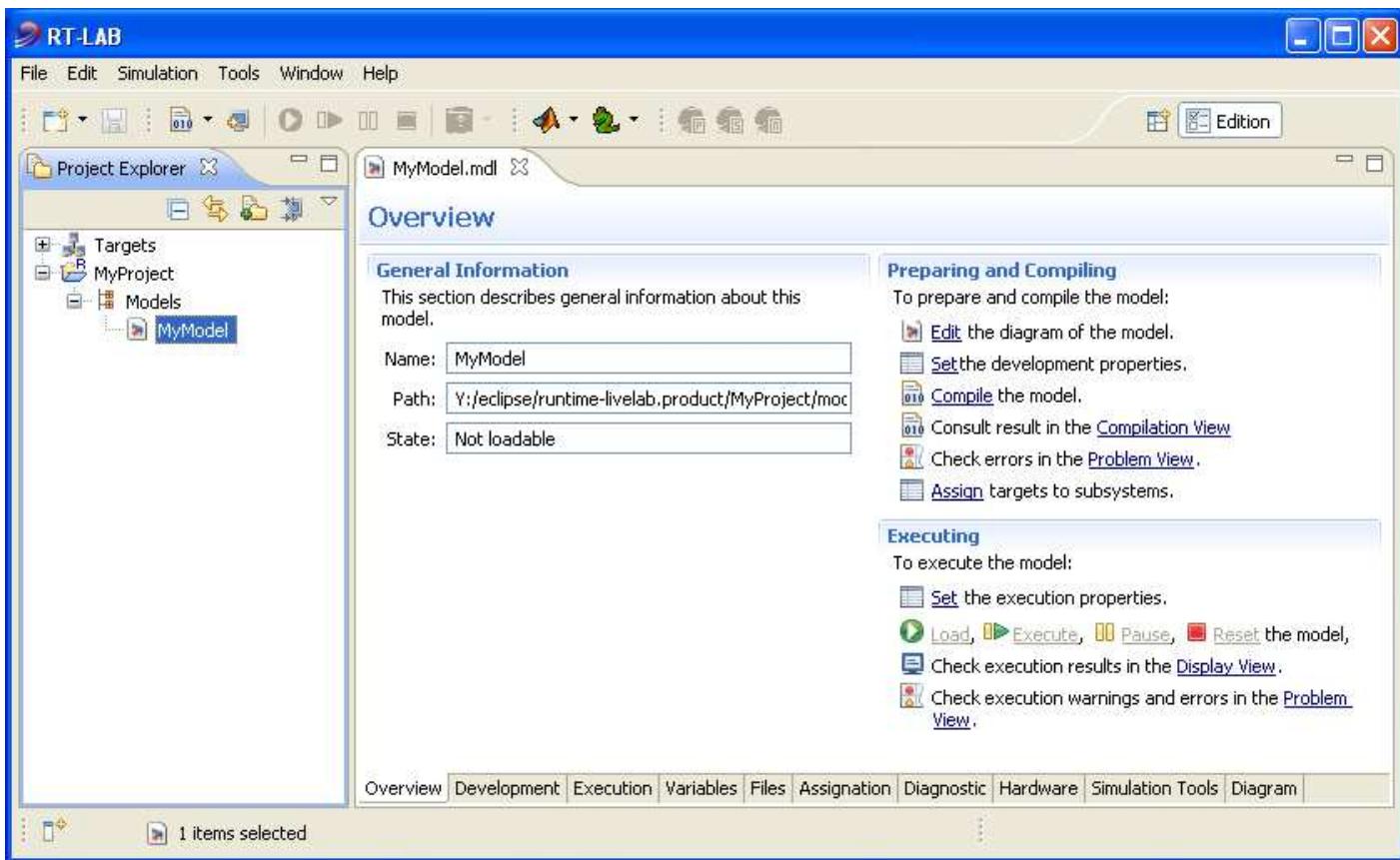
- Activate the Project Explorer view and select the project MyProject. From the view's popup menu choose **New > RT-LAB Model**.
- In the New Folder wizard, your project name appears by default in the **Project** field. This is because we chose to create the new model from your project's context menu.
- In the **Model name** field, type a unique name for your new model. Depending on the platform you are running on, some characters will not be allowed (for example, "MyModel").
- In the **Model type** field, select the type of model you want to create.
- In the **Model description** field, type a description of your model. Leave the box checked to use the default location for your new model.
- Leave the box checked to use the default location for your new project. Click **Next** when you are done.



- At this point you can select a template model containing a skeleton for your new model. Select the **Basic > model_1_subsystem** template. This template contains a model with one real-time subsystems and one console. This is a perfect model to run a simulation on 1 core target.



- Click **Finish** when you are done. The Project Explorer view will update to show your newly created model.

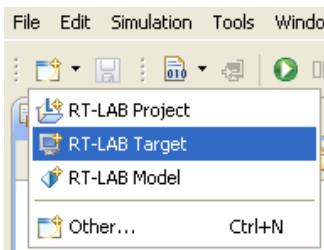


- The Workbench has an editor capable of editing properties of a model. A Model editor is automatically opened on the newly created model.

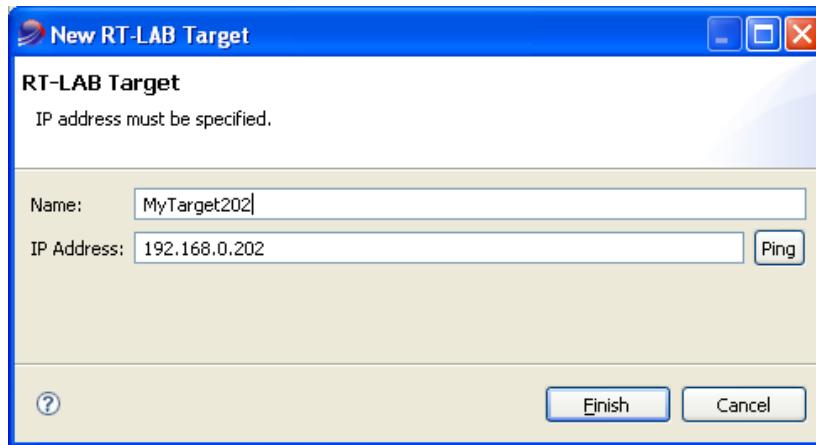
Using the New button

We have seen how to create resources using **File > New** and **New** from the context menu of one of the navigation views. We will now create a target using the third alternative, the toolbar **New** button.

- Select the target object in the project explorer view.
- In the Workbench window's toolbar, activate the drop-down menu on the New Wizard button and select **RT-LAB Target**. To activate the drop-down menu simply click on the down arrow.



- In the **Name** field, type a unique name for a new target computer. Do not use spaces or special characters in this name (for example, "MyTarget201"). It is a good practice to add the last number of the IP address of the target at the end of the name to help you to quickly identify your target.
- In the **IP address** field, type the IP address of your target computer. Use the following format for the IP address : **###.###.###.###**, where # is a number between 0 and 9.
- At this point, you can verify that the target is up and running by clicking the **Ping** button.



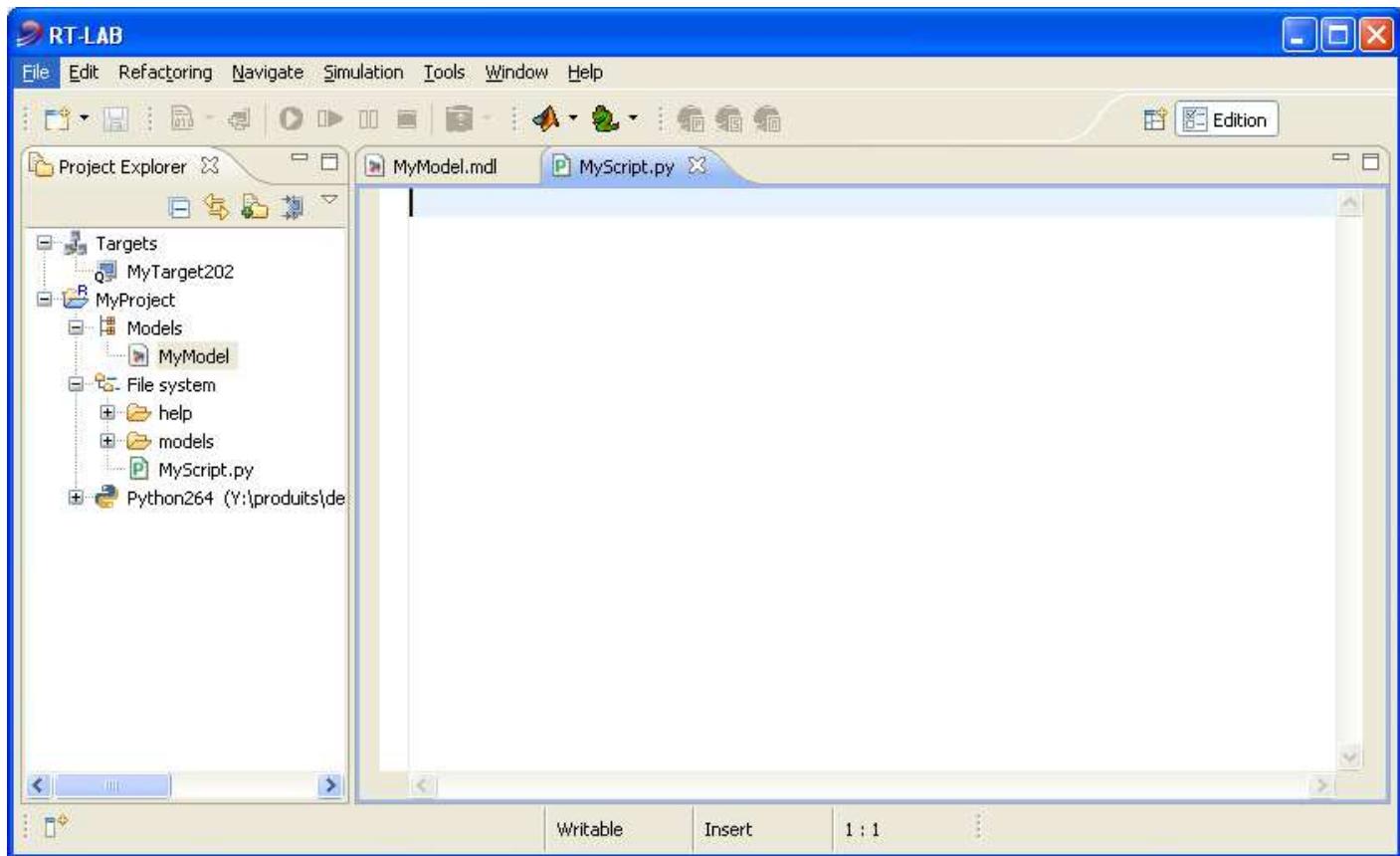
- Click **Finish** when you are done.
- In the Project Explorer view ensure your project is still selected and the view is active.
- Click **New Wizard** in the Workbench toolbar. Previously we clicked on the drop-down arrow of the New button. Here we clicked on the button itself which has the same effect as choosing **File > New > Other...**
- In the New wizard, select **General > File**. Then click Next.
- Once again, your project's name appears by default in the **Enter or select the parent folder** field.
- In the File name field, type a unique name for an .py file (Python script). Do not use any spaces or special characters in the file name (for example, "MyScript.py"). Click Finish when you are done.

- By default, the Project Explorer view do not display generic files and folders of the file system. Click the **Filter Resources** button of the Project Explorer toolbar to display the files and folders.



- You can easily turn it off by clicking the button again.

Now that we have created our resources, the Project Explorer view shows our project, the model, the target and the Python script. To the right of the Project Explorer view is the text editor open on the file we created (MyScript.py).



Preparing and building model

Now that there is a model in your project, here's how to prepare and build it for real-time simulation.

- [Preparing the build configuration](#)
- [Selecting the development node.](#)
- [Editing the model](#)
- [Building the model](#)
- [Assigning subsystems](#)

Selecting the development node.

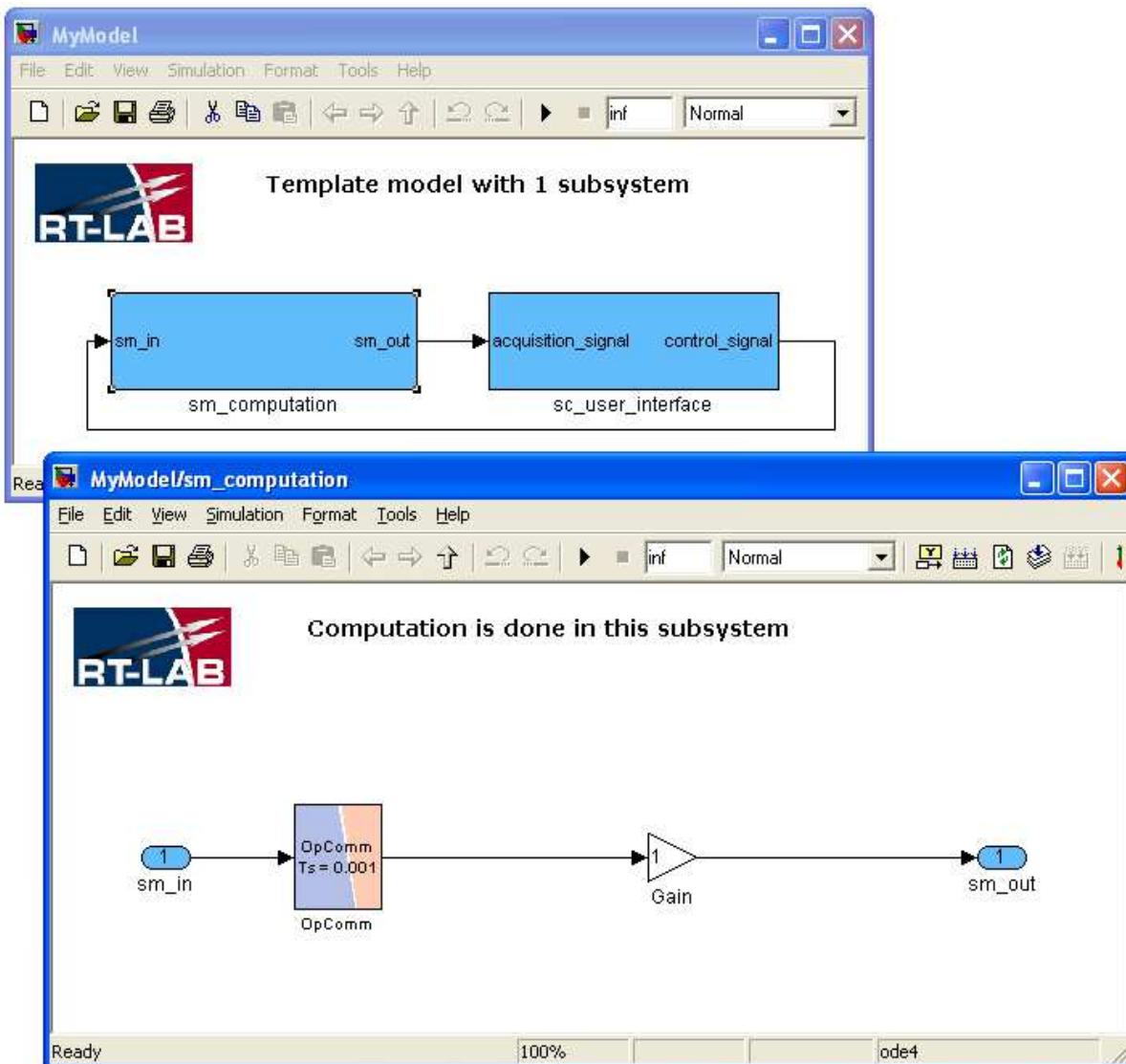
Before compiling your model for real-time simulation, you must select the target node on which your model will be compiled. This target node is called the development node. Normally this step is performed once, when configuring RT-LAB for the first time. To change the development node do the following.

- Activate the Project Explorer view.
- Select the target MyTarget in the Project Explorer view.
- To set this target as the development node, use one of the following methods:
 - From the view's popup menu choose **Set as development node** or
 - Open the RT-LAB preferences using the **Window > Preferences** menu, select the **RT-LAB -> Simulation Tools -> RT-LAB** preference page, and then select for the desired target platform the target node you want to use as development node.
- A small letter will appear on the target node indicating that this target node is used as development node.

Editing the model

Now that your model and RT-LAB are configured, here how to edit your model with RT-LAB.

- Activate the Project Explorer view.
- Select the model MyModel in the Project Explorer view.
- To edit your model, use one of the following methods:
 - From the the view's popup menu choose **Edit**
 - From the menu bar, select **Tools > Open MatLab**, drag and drop the model MyModel in MatLab command windows.
- Open sm_computation subsystem and add a Gain block between OpComm block and output of subsystem. Note that step correspond to the edition of your model.



- Save the modèle and close MatLab.

Preparing the build configuration

Before building your model for real-time, it is necessary to prepare the build configuration of the model.

- Activate the Project Explorer view.
- Select the model MyModel in the Project Explorer view.
- Double-click the model to open its editor.
- Select the Development page of the editor.
- Select the target platform (or operating system) corresponding to your target node.
- Note that the tabs of the Development page allow to configure the compiler and linker options.

Building the model

Now that your model and RT-LAB are configured, here how to build your model with RT-LAB.

- Activate the Project Explorer view.
- Select the model MyModel in the Project Explorer view.
- To start the complete build process, use one of the following methods:
 - From the menu bar, select **Simulation > Build**,
 - From the the view's popup menu choose **Simulation > Build** or
 - From the main toolbar, click on the build button.
- It is also possible to start the build process using the Build Configuration dialog. This dialog allows you to change RT-LAB configurations. For example, it allows to select the development node, the Matlab version used when compiling a Simulink model and the steps to perform while compiling. To open the Build Configuration dialog, do the following:
 - From the menu bar, select **Simulation > Build Configurations ...**,
 - From the the view's popup menu choose **Simulation > Build Configurations ...** or
 - From the main toolbar, select the drow down menu of the **Build** button, then click the **Build configurations ...** item menu.
- After starting the build process, the Compilation view will output compilation results.

Assigning subsystems

The last step before executing the model in real-time is to select on which target nodes each subsystems will be executed. This step is called “assignation”.

- Activate the Project Explorer view.
- Select the model MyModel in the Project Explorer view.
- Double-click the model to open its editor.
- Select the Assignment page of the editor.
- In the subsystems table, for each subsystem, select the **Assigned Node** cell and then select the desired target node from the drop down menu.

Executing model

Now that there is a builded model in your project, here's how to make a typical simulation (load, execute, change parameter, reset...).

- [**Selecting the simulation mode**](#)
- [**Loading the model**](#)
- [**Running the model**](#)
- [**Using the Console**](#)
- [**Resetting the model**](#)

Selecting the simulation mode

Before loading your model for real-time simulation, you must select the simulation mode of your model. You have choice between four simulation modes : Simulation, Simulation with low priority, Software Synchronized and Hardware Synchronized.

The model does not contain any I/O card, so you could choose Software Synchronized mode; the model will run in real-time and will be synchronized on internal timer. If you would have I/O card, we would choose Hardware Synchronized; in this mode the model would run in real-time and would be synchronized on external timer (for example synchronized on I/O card timer).

To set the simulation mode :

- Activate the Project Explorer view.
- Select the model MyModel in the Project Explorer view.
- Double-click the model to open its editor.
- Select the Execution page of the editor.
- Select Software Synchronized from Real-Time simulation mode.
- Note that the Execution page allow to configure the real-time options and performance options.

Loading the model

Now that your model and RT-LAB are configured, here how to load your model with RT-LAB.

- Activate the Project Explorer view.
- Select the model MyModel in the Project Explorer view.
- To start the load process, use one of the following methods:
 - From the menu bar, select **Simulation > Load**,
 - From the the view's popup menu choose **Simulation > Load** or
 - From the main toolbar, click on the load button.

Running the model

Now that your model are loaded, here how to execute your model with RT-LAB.

- Activate the Project Explorer view.
- Select the model MyModel in the Project Explorer view.
- To start the run process, use one of the following methods:
 - From the menu bar, select **Simulation > Execute**,
 - From the the view's popup menu choose **Simulation > Execute** or
 - From the main toolbar, click on the execute button.

Using the Console

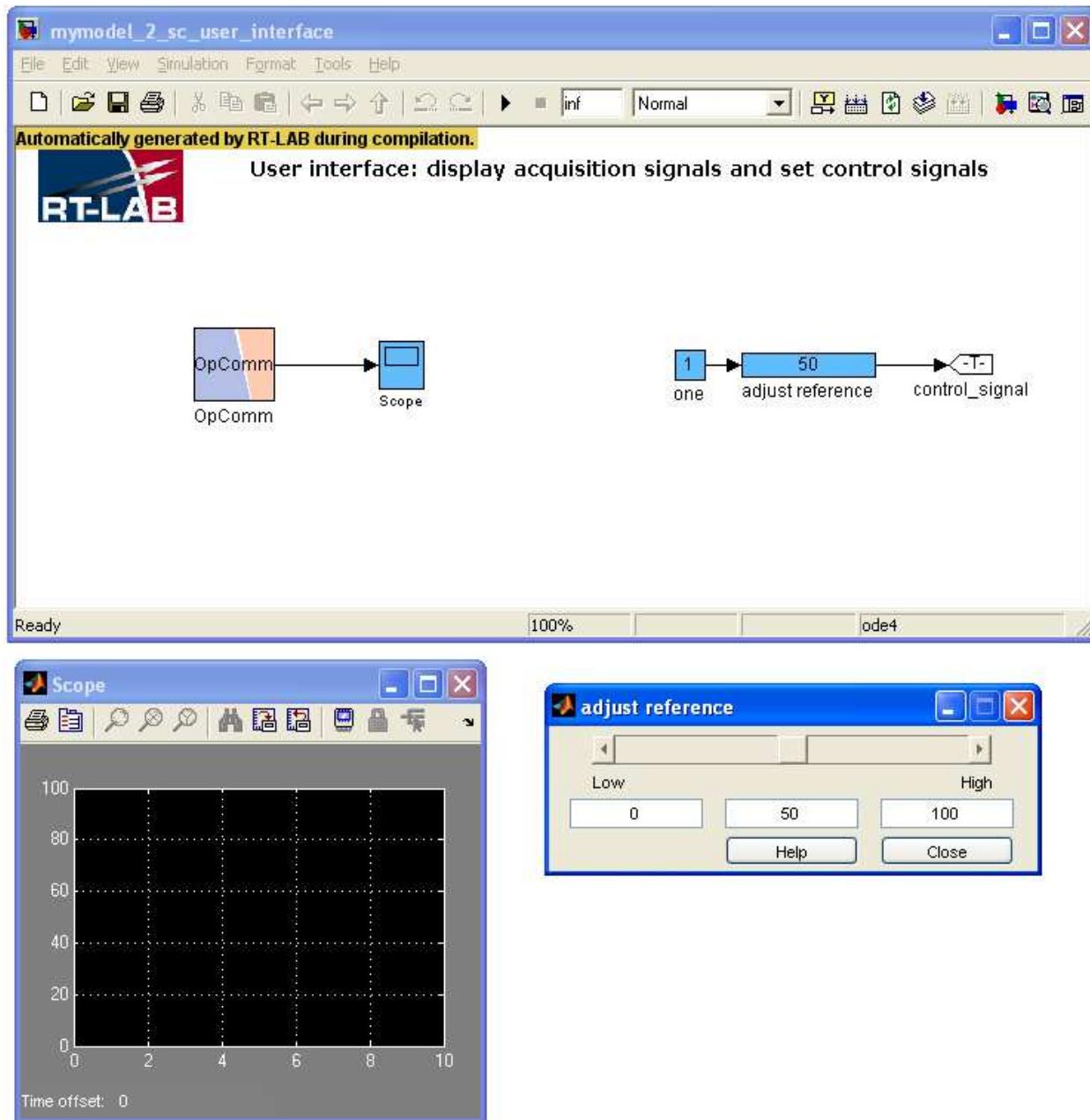
Now that your model and RT-LAB is running, here how to interact with your model by using the Simulink Console.

- Activate the Project Explorer view.
- Select the model MyModel in the Project Explorer view.
- Under the model MyModel, the list of MyModel subsystems appears



- To open the Simulink Console, use one of the following methods :
 - Select and double-click on the subsystem corresponding to the console subsystem. (For your model : double-click on sc_user_interface subsystem) or
 - Before the model loading step, double-click on the model to open its editor, select the Simulation tools page and check Handle Console automatically.

The Simulnik Console should be open :



- To start the Simulink Console, press Start Simulation in MatLab. You can interact with the model through the Simulink Console by changing Adjust reference, and observe effect your change in the Simulink scope.

Resetting the model

When you finish the simulation of your model, here how to reset your model with RT-LAB.

- Activate the Project Explorer view.
- Select the model MyModel in the Project Explorer view.
- To start the reset process, use one of the following methods:
 - From the menu bar, select **Simulation > Reset**,
 - From the the view's popup menu choose **Simulation > Reset** or
 - From the main toolbar, click on the reset button.
- Close the Simulink Console.

Closing and editor

Now that there is an editor open, here's how to close it.

- Select the MyScript.py editor tab.
- To close the editor, choose one of the following options:
 - Click the close button ("X") in the tab of the editor.



- Select **File > Close** from the menu bar.
- Note the prompt to save the file before the editor is closed.
- Click OK to save any changes and close the editor.

If the editor was closed using **File > Close**, notice that the option **File > Close All** was also displayed. This is a quick way to close all of the open editors. If **File > Close All** is chosen, a prompt will appear to choose which editors with unsaved changes should be saved.

Navigating resources

This section will work with the Project Explorer view. This view is initially part of the RT-LAB Edition perspective. To experiment with other views, they can be displayed by using the **Window > Show View** menu.

One important view to become familiar with is the Project Explorer view, which displays information about the contents of the Workbench and how the resources relate to each other in a hierarchy.

In the Workbench, all resources reside in projects. Projects can contain Model, folders and/or individual files.

Opening resources in the Project Explorer

Using the Project Explorer there are several ways to open an editor.

- In the Project Explorer view select the file MyScript.py
- To open an editor on the file choose one of the following approaches:
 - To edit a resource using the default editor for that resource, either double click on the resource, or select the resource and choose **Open** from its popup menu.
 - To use a specific editor to edit the resource, start by selecting the resource, and choose the Open With option from the popup menu.
- The Workbench remembers the last editor that was used for editing a specific file. This makes it easier to use the same editor down the road.

The default editors can be configured using the General > Editors > File Associations preference page.

Using the Project Explorer there is also a simple way to open an editor of a model.

- In the Project Explorer view select the model MyModel.
- To open an editor on the model double click on the resource, or select the resource and choose **Open** from its popup menu.
- To edit the block diagram of the model using Simulink or EMTP-Works, select the resource and choose **Edit** from its popup menu.

Files

The project, model and file that you create with the Workbench are all stored under a single directory that represents your workspace. The location of the workspace was set in the dialog that first opens when you start the Workbench.

If you have forgotten where that location is, you can find it by selecting **File > Switch Workspace....** The workspace directory will be displayed in the dialog that appears. **IMPORTANT:** After recording this location, hit Cancel to close the dialog, or the Workbench will exit and re-open on whatever workspace was selected.

All of the project, model and file that you create with the Workbench are stored as normal directories and files on the machine. This allows the use of other tools when working with the files. Those tools can be completely oblivious to the Workbench. A later section will look at how to work with external editors that are not integrated into the Workbench.

Deleting resources

Here are instructions on how to delete the Python script file.

- Select file MyScript.py in the Project Explorer view.
- To delete the file do one of the following:
 - From the project's pop-up menu choose **Delete**
 - Press the DEL key
 - Choose **Edit > Delete** from the main menu
- A prompt will ask for confirmation of the deletion. Accept and click **Yes**.

The same steps work for any resource shown in the Project Explorer view except for project resource.

When deleting a project, it is also possible to delete the project from the workspace but keep its contents in the file system.

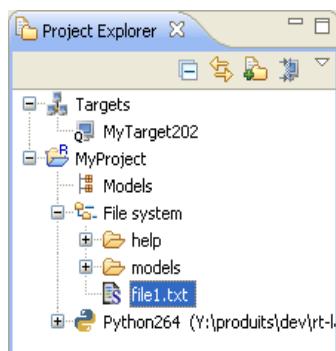
Copying and renaming

Workbench resources can be copied and renamed using popup menu operations in the Project Explorer view. In this section files that have been created will be copied and renamed.

Prior to copying files, some setup is required:

Setup

- In the Project Explorer view, create a new text file file1.txt. The Project Explorer view should look like this:



- Double click on file1.txt and ensure that it contains the following text.

*This is sample text file.
There is not much else
we can really say about it other
than it has five lines of
text that are rather dull.*
- Close the editor on file1.txt file.

Copying

You can copy file1.txt to the help folder using the following steps.

- Ensure that the setup described in the introduction to this section has been performed.
- In the Project Explorer view, select file1.txt.
- From the file's context menu, select **Copy** (or Ctrl+C)
- In the Project Explorer view, select help as the destination.
- From the folder's context menu, select **Paste** (or Ctrl+V).

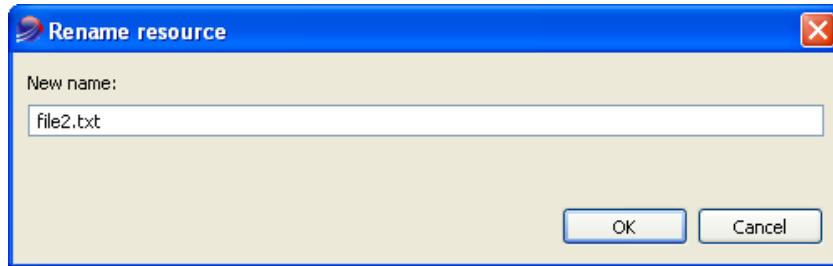
As an alternative to copying files using the copy operation, it is also possible to copy files by holding down the Ctrl key while dragging a file from one folder to another folder.

Once the file has been copied it can be renamed.

Renaming

Now that file1.txt has been copied from MyProject folder to Help folder it is ready to be renamed as something else.

- In the Project Explorer view, select file1.txt in help folder.
- From the file's context menu, select **Rename**.
- The navigation view overlays the file's name with a text field. Type in file2.txt and press Enter.



- To halt the renaming of a resource, Escape can be pressed to dismiss the text field.

Copy and rename works on folders as well.

- In the Project Explorer view, select the folder Help.
- From the folder's context menu choose **Rename**.
- Once again the navigation view overlays the folder name with an entry field to allow the typing in of a new name. Change the folder name to be HelpFolder.
- Rename the folder back to its original name (Help).

Searching

RT-LAB objects, text strings and files can be searched within the Workbench. Search dialogs can be opened by selecting the search menu or by clicking on the Search button in the main toolbar.

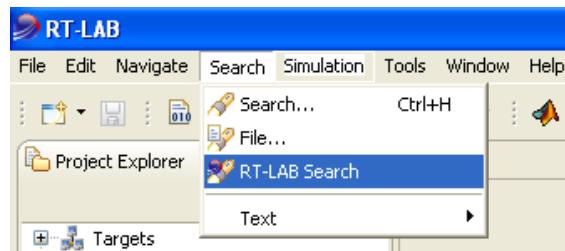


Figure 1:Search Menu

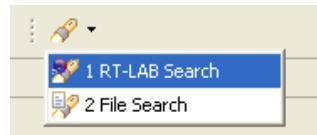


Figure 2:Search buttons - Main toolbar

The Ctrl+H shortcut key can be used to open the RT-LAB Search dialogs.

The search dialogs available in the workbench are:

- **RT-LAB Search**
- **File Search**

The results of the search will be displayed in the **Search View**.

It is also possible to start a quick text search. To perform this type of search, select a text, select the sub menu Text from the main menu Search and then click on Workspace. It will start a file search with the specified text and the results will be displayed in the **Search View**.

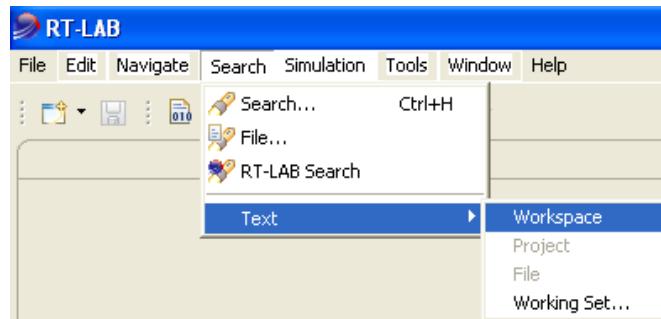


Figure 3:Quick text search

Rearranging views and editors

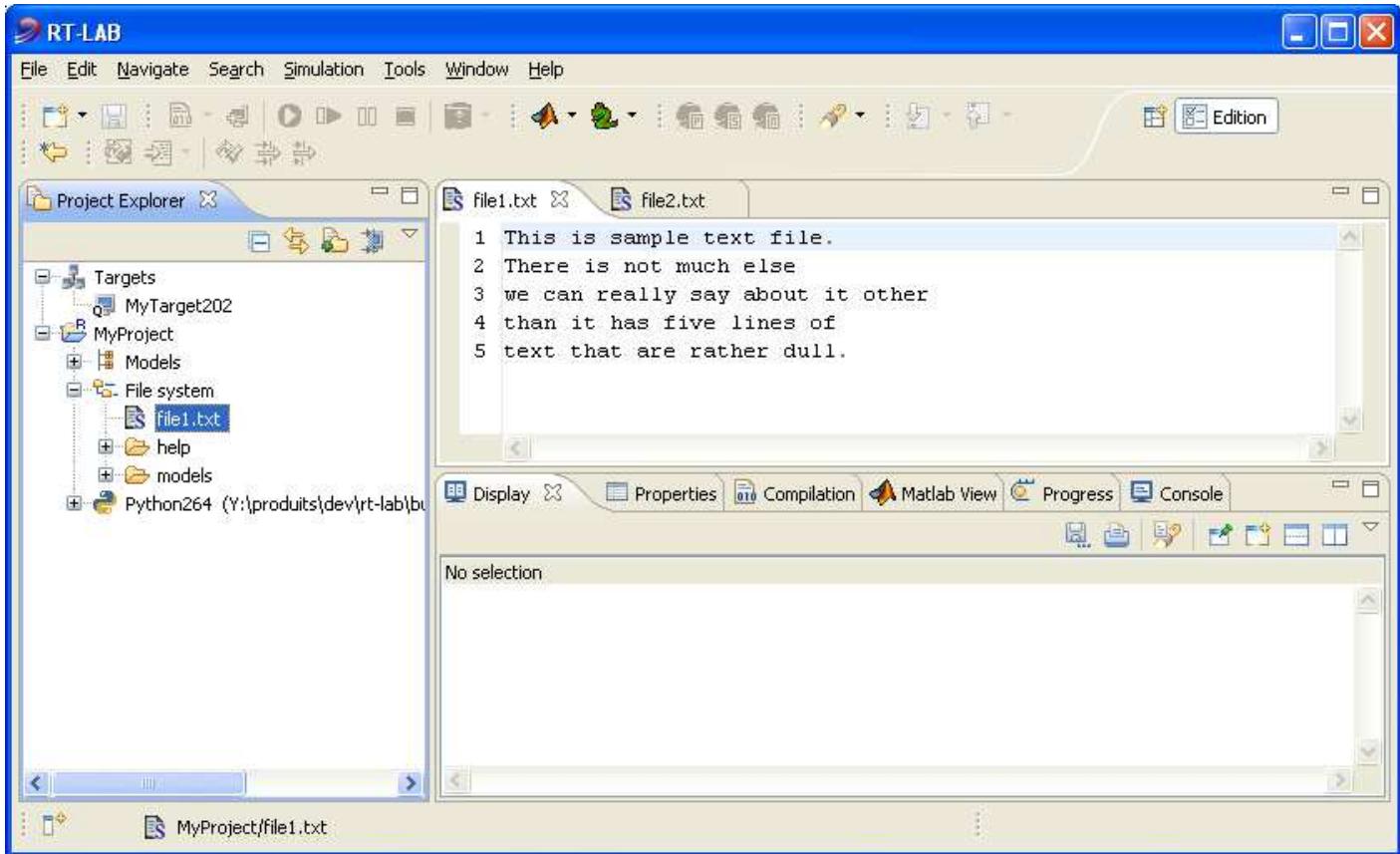
This section will explain how to rearrange editors and views to customize the layout of the Workbench.

Setup

Before rearranging the Workbench, a little housekeeping is required.

- Start by choosing **Window > Reset Perspective** and selecting **OK**. This will reset the current perspective to its original views and layout.
- Ensure there are editors open for file1.txt and file2.txt. Close any other editors.

The Workbench should now look like this:



Drop cursors

Drop cursors indicate where it is possible to dock views in the Workbench window. Several different drop cursors may be displayed when rearranging views.

ICON	DESCRIPTION
↑	Dock above If the mouse button is released when a dock above cursor is displayed, the view will appear above the view underneath the cursor.
↓	Dock below If the mouse button is released when a dock below cursor is displayed, the view will appear below the view underneath the cursor.
→	Dock to the right If the mouse button is released when a dock to the right cursor is displayed, the view will appear to the right of the view underneath the cursor.
←	Dock to the left If the mouse button is released when a dock to the left cursor is displayed, the view will appear to the left of the view underneath the cursor.
📁	Stack If the mouse button is released when a stack cursor is displayed, the view will appear as a tab in the same pane as the view underneath the cursor.
🚫	Restricted If the mouse button is released when a restricted cursor is displayed, the view will not dock there. For example, a view cannot be docked in the editor area.

Rearranging views

The position of the Project Explorer (or any other) view in the Workbench window can be changed.

- Click in the title bar of the Project Explorer view and drag the view across the Workbench window. Do not release the mouse button yet.
- While still dragging the view around on top of the Workbench window, note that various drop cursors appear. These drop cursors (see previous section) indicate where the view will dock in relation to the view or editor area underneath the cursor when the mouse button is released. Notice also that a rectangular highlight is drawn that provides additional feedback on where the view will dock.
- Dock the view in any position in the Workbench window, and view the results of this action.
- Click and drag the view's title bar to re-dock the view in another position in the Workbench window. Observe the results of this action.
- Click and drag the view's title bar outside the Workbench window. Notice that it becomes a "Detached" window (hosted in its own shell).
- Drag the Project Explorer view over the Compilation view. A stack cursor will be displayed. If the mouse button is released the Project Explorer will be stacked with the Compilation view into a tabbed notebook. Please note that you must grab the view tab, **NOT** the window title. This is used only for repositioning the window containing the detached view.
- Finally, drag the view to its original location.

Rearranging tabbed views

In addition to dragging and dropping views on the Workbench the order of views can also be rearranged within a tabbed notebook.

- Choose **Window > Reset Perspective** to reset the RT-LAB Edition perspective back to its original layout.
- Click on the Compilation View title bar and drag it on top of the Project Explorer view. The Compilation View will now be stacked on top of the Project Explorer view.
- Click the tab of the Project Explorer view and drag it to the right of the Compilation View tab.



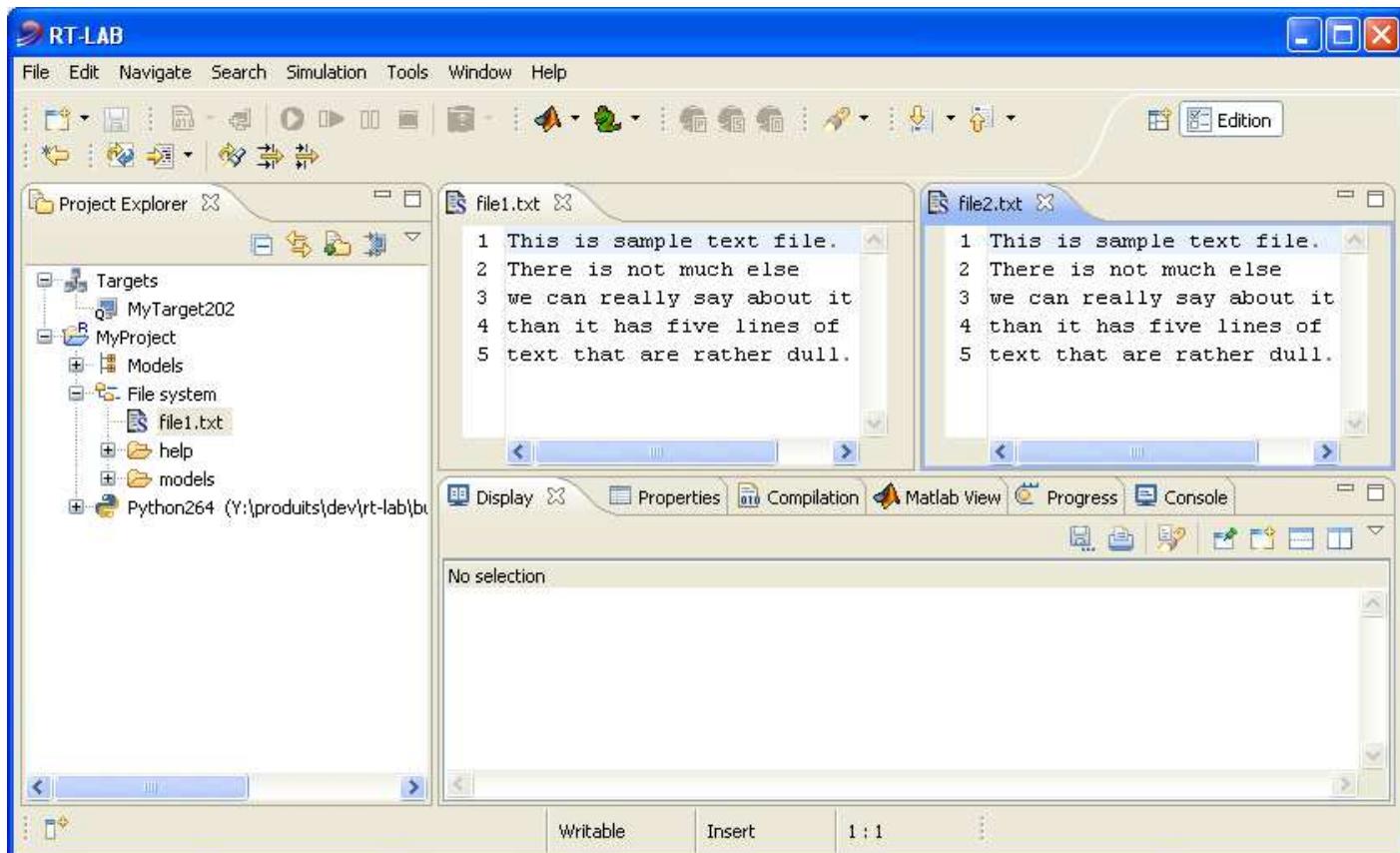
- Once the cursor is to the right of the Compilation tab and the cursor is a stack cursor release the mouse button. Observe the Project Explorer tab is now to the right of the Compilation View tab.



Tiling editors

The Workbench allows for the creation of two or more sets of editors in the editor area. The editor area can also be resized but views cannot be dragged into the editor area.

- Open at least two editors in the editor area by double-clicking editable files in one of the navigation views.
- Click and drag one of the editor's tabs out of the editor area. Do not release the mouse button.
- Notice that the restricted cursor displays if an attempt is made to drop the editor either on top of any view or outside the Workbench window.
- Still holding down the mouse button, drag the editor over the editor area and move the cursor along all four edges as well as in the middle of the editor area, on top of another open editor. Notice that along the edges of the editor area the directional arrow drop cursors appear, and in the middle of the editor area the stack drop cursor appears.
- Dock the editor on a directional arrow drop cursor so that two editors appear in the editor area.
- Notice that each editor can also be resized as well as the entire editor area to accommodate the editors and views as necessary.
- It is important to observe the color of an editor tab (in the figure below there are two groups, one above the other)
 - blue** - indicates that the editor is currently active
 - default** (gray on Windows XP) - indicates that the editor was the last active editor. If there is an active view, it will be the editor that the active view is currently working with.
- Drag and dock the editor somewhere else in the editor area, noting the behavior that results from docking on each kind of drop cursor. Continue to experiment with docking and resizing editors and views until the Workbench has been arranged to satisfaction. The figure below illustrates the layout if one editor is dragged and dropped below another.



Maximizing and minimizing elements of the workbench presentation

RT-LAB presentation provides a rich environment consisting of (in its basic form) an Editor Area (containing one or more stacks showing the open editors) surrounded by one or more View Stacks (each containing one or more views). These various parts compete for valuable screen real-estate and correctly managing the amount of screen given to each can greatly enhance your productivity within the IDE.

The two most common mechanisms for managing this issue are 'minimize' (i.e. make me use as little space as possible) and 'maximize' (i.e. give me as much space as you can). The RT-LAB presentation provides a variety of ways to access these operations:

- Using the minimize and maximize buttons provided on a stack's border
- Selecting the 'Minimize' or 'Maximize' item on the context (right-click) menu for a stack
- Double-clicking on a stack
- Using 'Ctrl + M': this is a key binding for a command that will toggle the currently active part between its 'maximized' and its 'restored' (i.e. normal) states.

Maximize

It is desirable at times to focus your attention on one particular part to the exclusion of the others. The most popular candidate for this is, of course, maximizing the editor area in order to make as much of the display available for editing as possible (but there are workflows where it would make sense to focus on a view as well).

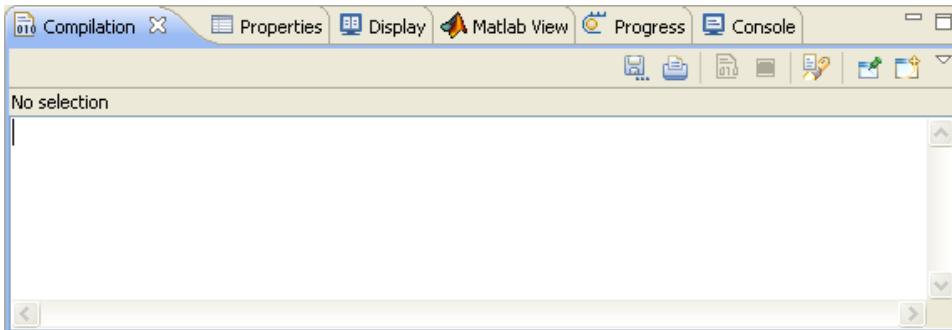
The default presentation implements the maximize behavior by minimizing all stacks except the one being maximized. This allows the maximized stack to completely occupy the main presentation while still allowing access any open views in your perspective by using the icons in their Trim Stack (the area around the edges of the window is called the 'trim').

The behavior for managing the editor maximization operate on the complete Editor Area (rather than simply maximizing the particular Editor Stack. This allows for 'compare' workflows which require the ability to see both files in a split editor area at the same time.

Minimize

Another way to optimize the use of the screen area is to directly minimize stacks that are of no current interest. The default presentation minimizing a stack will cause it to be moved into the trim area at the edges of the workbench window, creating a Trim Stack. View Stacks get minimized into a trim representation that contains the icons for each view in the stack.

This view stack



becomes this Trim Stack when minimized



The minimize behavior for the Editor Area is somewhat different; minimizing the Editor Area results in a trim stack containing only a placeholder icon representing the entire editor area rather than icons for each open editor (since in most cases all the icons would be the same, making them essentially useless).

The editor area



becomes this Trim Stack when minimized



If your particular workflow is such that you need to have more than one element (i.e. having the Editor Area and a View Stack in the presentation at the same time) you can still gain additional screen space by minimizing the stacks that aren't of current interest. This will remove them from the main presentation and place them on the outer edge of the workbench window as Trim Stacks, allowing more space for the remaining stacks in the presentation.

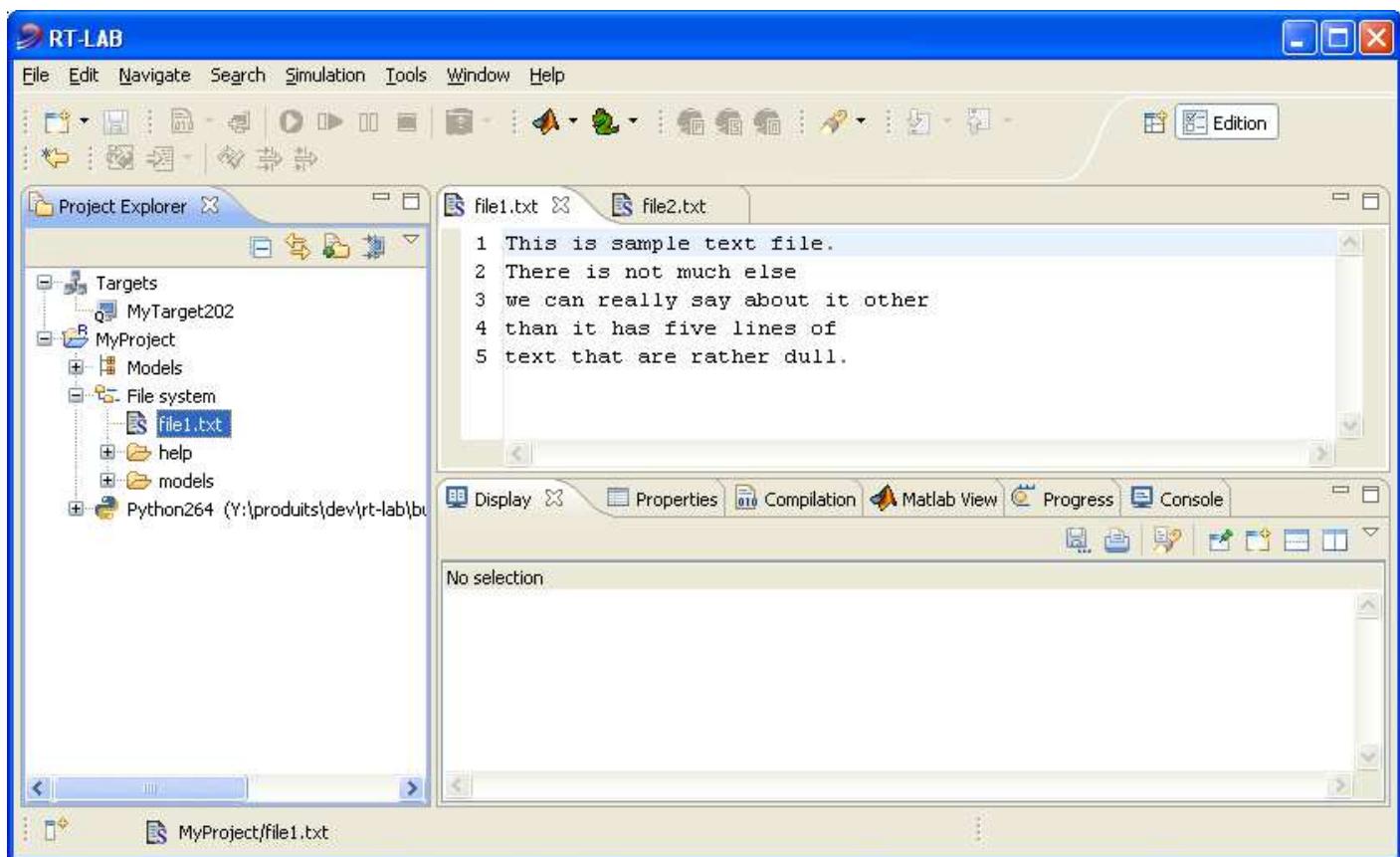
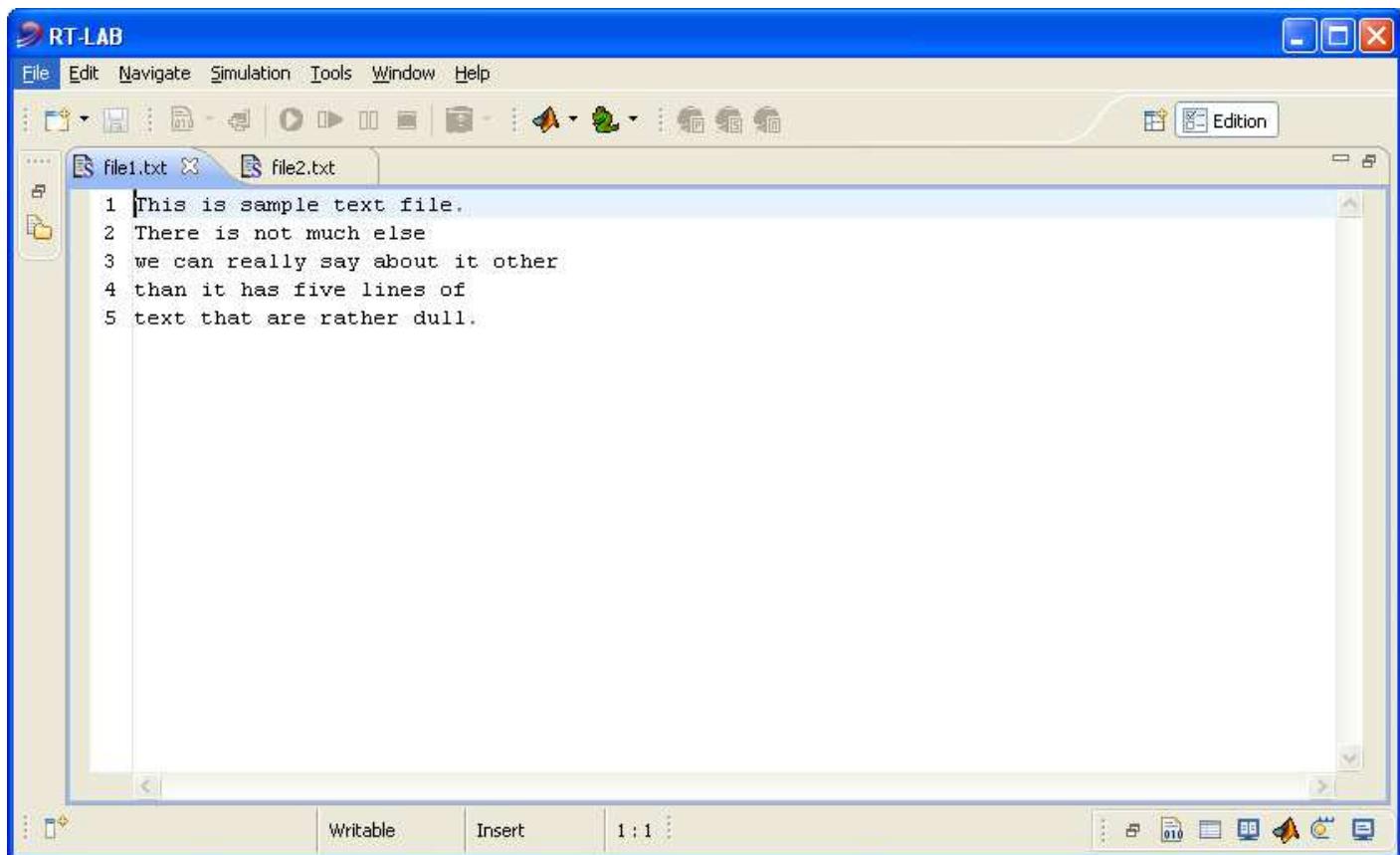
Note: There are two ways to end up with a stack in the trim:

- Directly minimizing the stack
- As the result of another stack being maximized

Depending on how the Trim Stack was created its behavior is different; when un-maximizing only those trim stacks that were created during the initial maximize will be restored to the main presentation while stacks that were independently minimized stay that way.

Tip: This difference is important in that it allows you fine grained control over the presentation. While using maximize is a one-click operation it's an 'all or nothing' paradigm (i.e. no other stack is allowed to share the presentation with a maximized stack). While adequate for most tasks you may find yourself wanting to have the presentation show more than stack. In these scenarios don't maximize; minimize all the other stacks except the ones you want in the presentation. Once you have it set up you can still subsequently maximize the editor area but the un-maximize will only restore the particular stack(s) that were sharing the presentation, not the ones you've explicitly minimized.

Normal Presentation

**Editor Area Maximized**

Perspectives

A perspective defines the initial set and layout of views in the Workbench window. One or more perspectives can exist in a single Workbench window.

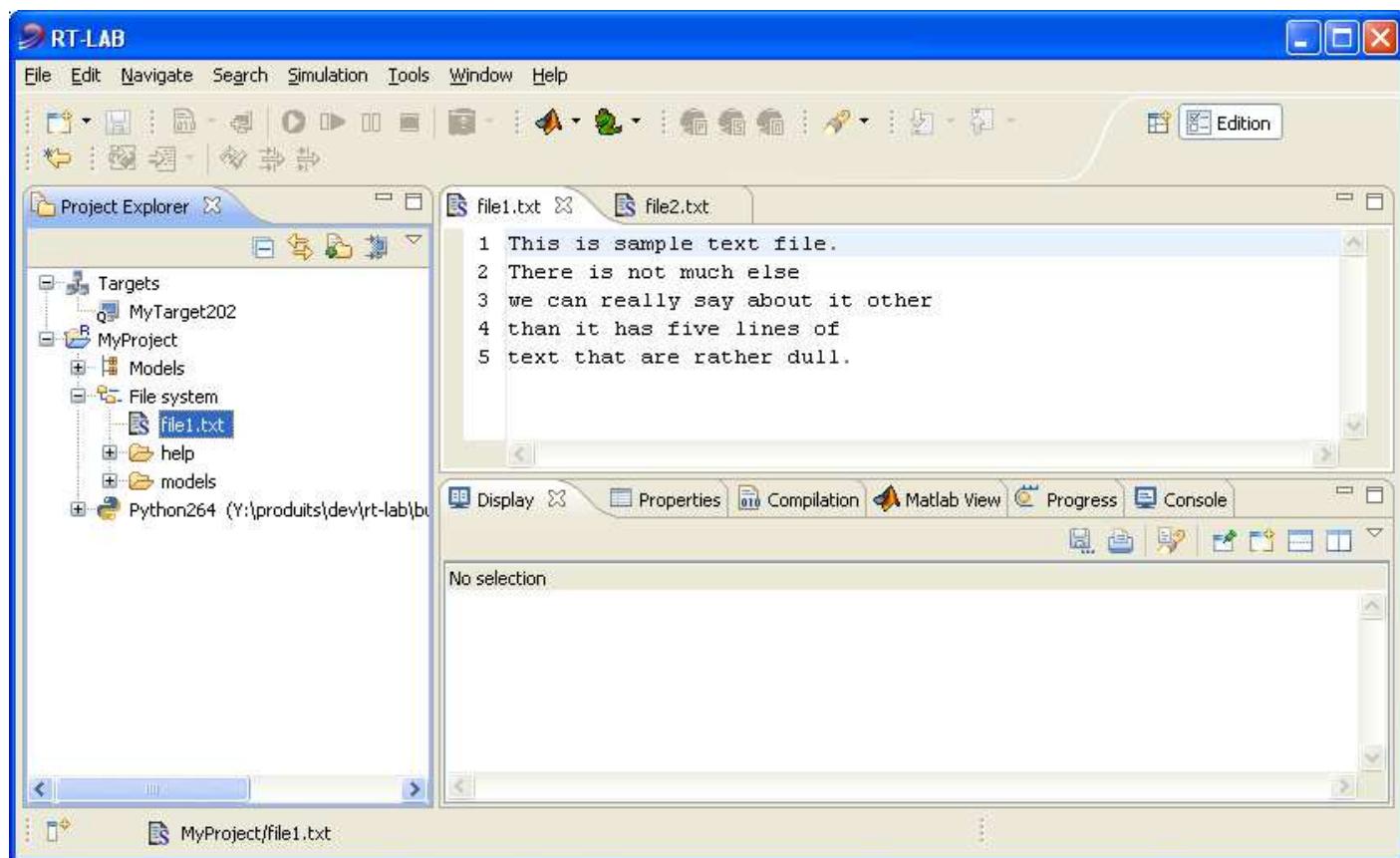
Perspectives can be opened in the same (existing) Workbench window.

Perspectives define visible action sets, which can be changed to customize a perspective. A perspective that is built in this manner can be saved, creating a custom perspective that can be opened again later.

The Workbench window displays one or more perspectives. RT-LAB determines initially what default perspective is displayed, in this example it is the RT-LAB Edition perspective. A perspective consists of views such as the Project Explorer as well as editors for working with resources. More than one Workbench window can be open at any given time.

So far, only the RT-LAB Edition perspective (shown below) has been used in this tutorial. This section will explore how to open and work with other perspectives.

A perspective provides a set of functionality aimed at accomplishing a specific type of task, or working with a specific type of resource.



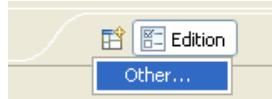
New perspectives

There are several ways to open a new perspective within this Workbench window:

- Using the **Open Perspective** button on the shortcut bar.
- Choosing a perspective from the **Window > Open Perspective** menu.

To open one by using the shortcut bar button:

- Click on the Open Perspective button .



- A menu appears showing the same choices as shown on the **Window > Open Perspective** menu. Choose Other from the menu.
- In the Select Perspective dialog choose another perspective and click **OK**. The perspective is displayed.
- There are several other interesting things to take note of.
 - The title of the window now indicates that the new perspective is in use.
 - The shortcut bar contains several perspectives, the original Edition perspective, the new perspective and a few others. The new perspective button is pressed in, indicating that it is the current perspective.
 - To display the full name of the perspective right click the perspective bar and check **Show Text**.
- In the shortcut bar, click on the Edition perspective button. The Edition perspective is once again the current perspective. Notice that the set of views is different for each of the perspectives.

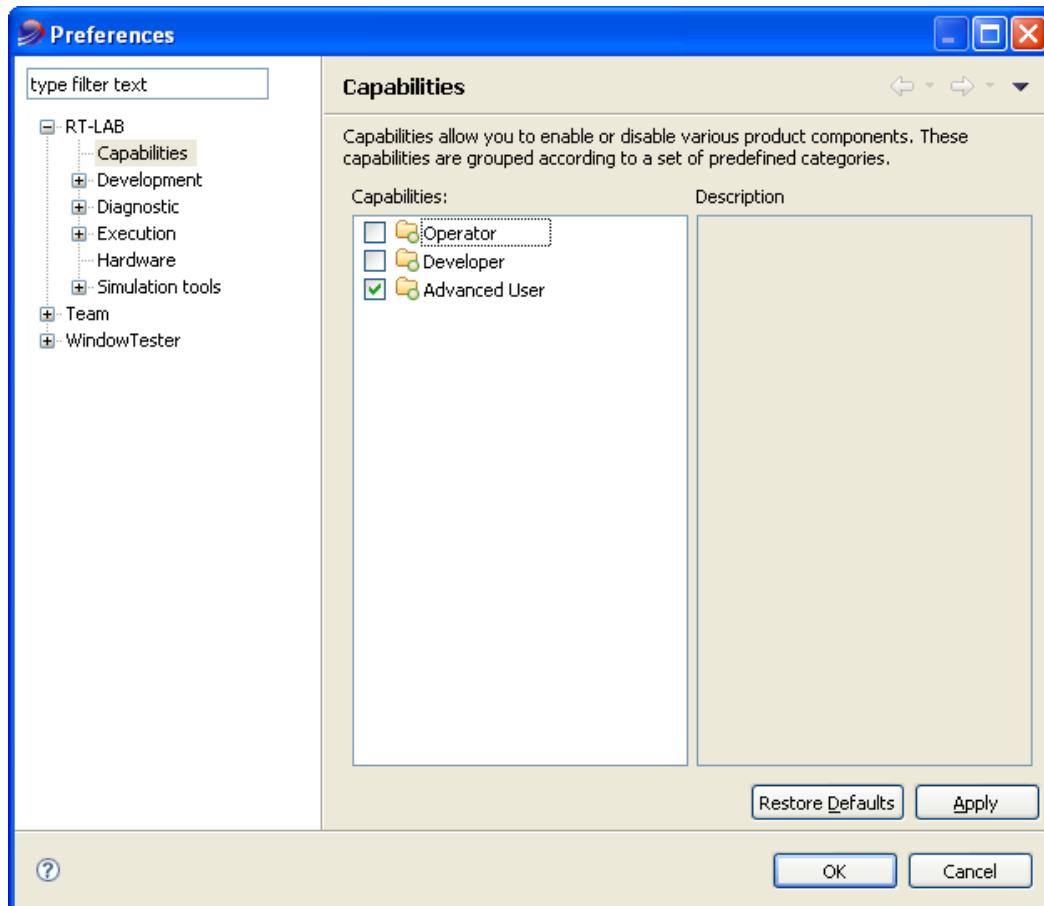
Saving perspectives

This tutorial has demonstrated how to add new views to the perspective, rearrange the views and convert views into fast views. The Workbench also allows this layout to be saved for future use.

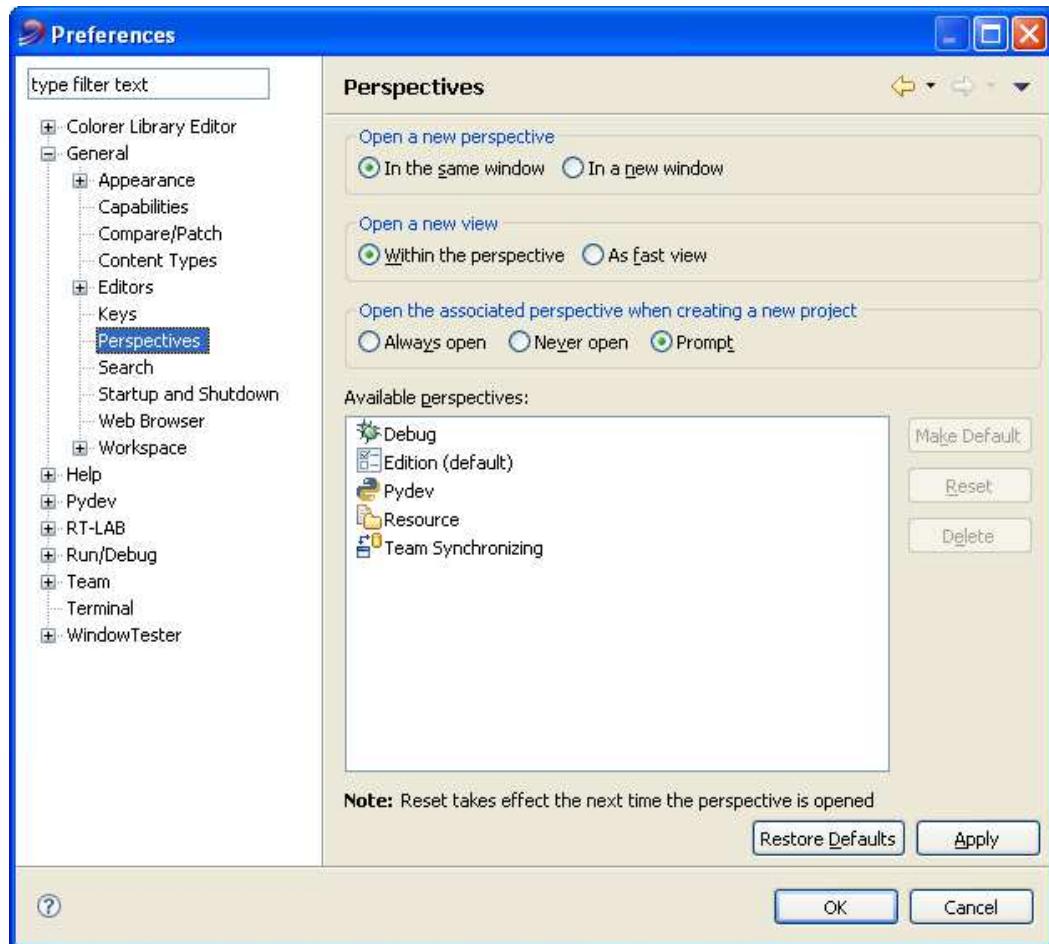
- In the shortcut bar click on the Edition perspective. The Edition perspective is now active.
- Drag the Compilation view and stack it with the Project Explorer view.
- Choose **Window > Save Perspective As...**
- The Save Perspective As dialog allows for an existing perspective to be redefined or for a new perspective to be created. Click **OK** to update the Edition perspective and **Yes** to the subsequent confirmation dialog. The new perspective layout will be used if the perspective is reset or if a new one is opened.
- In the Edition perspective move the Compilation view so that it is now stacked with the Display view.
- Choose **Window > Reset Perspective**. Notice the Compilation view is stacked with the Project Explorer view. Originally when the Workbench was first started it was stacked with the Display view, but because the perspective was saved with the Project Explorer view stacked, it now considers this its initial layout.

While the Edition perspective has been changed, there is a way to get back the original layout. To reset the Edition perspective to its original layout:

- Choose **Window > Preferences**.
- Expand **RT-LAB** and select **Capabilities**.
- Select the **Advanced** capability and click **OK**



- Choose **Window > Preferences**.
- Expand **General** and select **Perspectives**.
- Select **Edition (default)** > **Make Default** and then click **OK**.



- Any changes to the saved state of the perspective has now been undone. To update the current copy of the Resource perspective that is being worked with, also choose **Window > Reset Perspective** from the Workbench's menu bar.
- Expand **RT-LAB** again and select **Capabilities**.
- Unselect the **Advanced** capability and click **OK**.

Configuring perspectives

In addition to configuring the layout of a perspective you can also control several other key aspects of a perspective. These include:

- The **New** menu.
- The **Window > Open Perspective** menu.
- The **Window > Show View** menu.
- Action sets that show up on the toolbar.

Try customizing one of these items.

- In the shortcut bar click on the Edition perspective.
- Select **Window > Customize Perspective...**
- Select the **Commands** tab.
- Uncheck the Old tools and click OK.
- Observe that the toolbar now exclude buttons for launching old RT-LAB application.
- After experimenting with the other options on the Customize Perspective dialog, choose **Window > Reset Perspective** to return the perspective to its original state.

Comparing

The Workbench allows for the comparison of multiple resources and for the presentation of the results in a special compare editor.

Setup

Before commencing with compare a few files must be created. This will also be a good time to recap some of the basic features that have already been introduced.

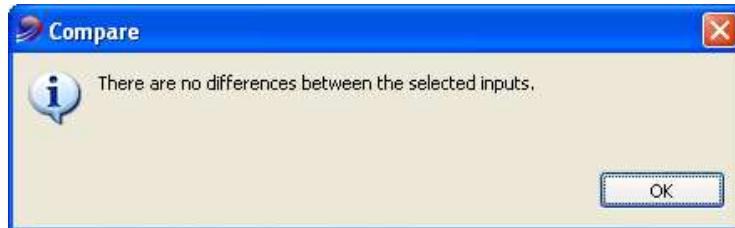
- Start by selecting the file1.txt in the Project Explorer view. Double-click on it, to open the text editor.
- In the editor for file1.txt type the following lines of text and save the file:
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
- In the Project Explorer view select file1.txt and use Ctrl+C to copy the file.
- Use Ctrl+V (Paste) to create the copy. In the name conflict dialog which appears, rename the file to file2.txt.

There are now two identical files, file1.txt and file2.txt.

Simple compare

In the Project Explorer view select file1.txt and file2.txt choose **Compare With > Each Other** from the context menu.

A dialog will appear indicating that the two files are the same.



Edit file1.txt as follows:

- delete line 1 "*This is line 1.*"
- change line 3 to be "*This is a much better line 3.*"
- insert a line 4a (before line 5) that reads "*This is line 4a and it is new*"

The file (file1.txt) should now read as follows:

This is line 2.

This is a much better line 3.

This is line 4.

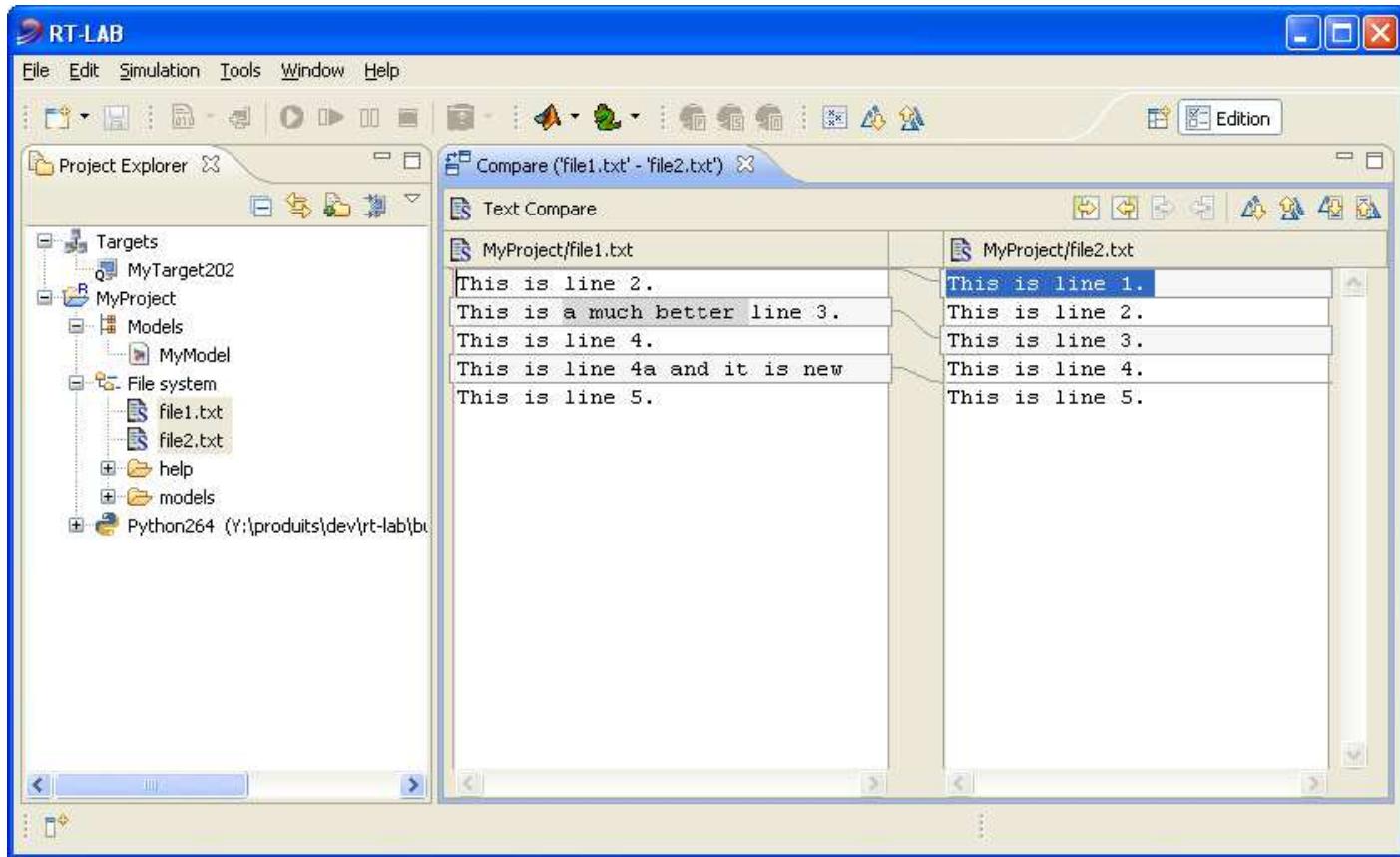
This is line 4a and it is new

This is line 5.

Save the contents of the file by choosing **File > Save** (or pressing Ctrl+S).

To compare the files, once again select file1.txt and file2.txt and choose **Compare With > Each Other** in the navigation view context menu.

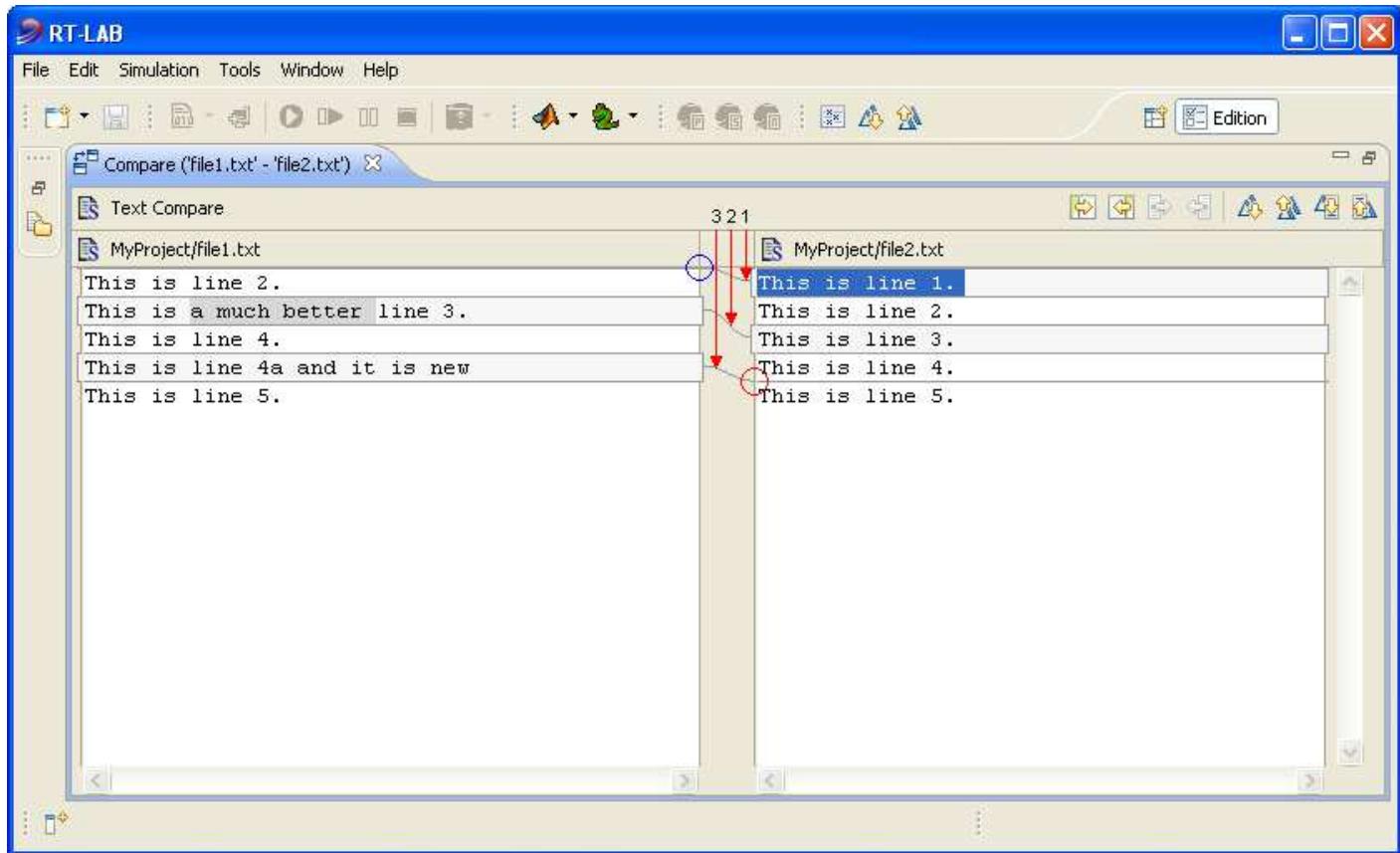
A special compare editor opens. The next section will explain how to use this compare editor.



Understanding the comparison

Comparing file1.txt and file2.txt resulted in the following compare editor. The left side shows the contents of file1.txt and the right side shows the contents of file2.txt. The lines connecting the left and right panes indicate the differences between the files.

If more room is needed to look at the comparison, the editor tab can be double clicked to maximize the editor.



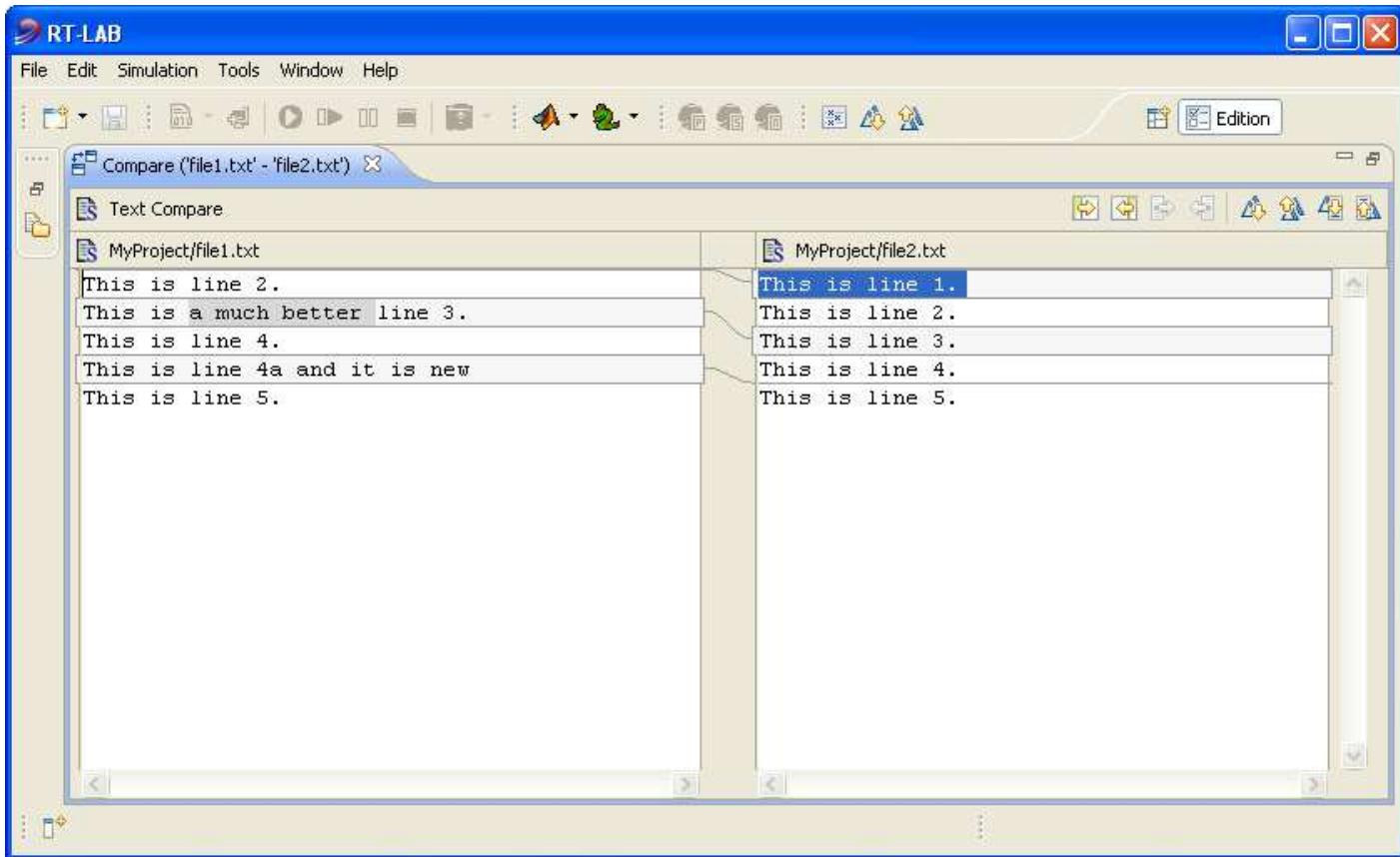
The numbered changes on the left side of the difference editor are as follows:

- Starting with the top line (in the left pane) the difference bar (in the area of the blue circle) indicates something is missing from the very top of the left file. Follow the difference band (see #1) to the right file. It contains "This is line 1".
- The next line "This is line 2." is white indicating it matches the right file.
- Moving onto the next line (colored in the background color), see that the left file and right file have different contents for this line (see #2).
- The next line (This is line 4) is once again in white, so it can be skipped.
- The next line exists in the left file but since it is in the background color its difference bar can be followed to the right (see #3) and notice that the right file does not contain the line (see red circle).

Initially the compare editor might seem a bit daunting but when simply working down the left side and focusing on the items marked as gray, and those items missing from the left side, it turns out not to be as tricky as it first seems.

Working with comparison

Comparing file1.txt and file2.txt resulted in the following compare editor. This section demonstrates how to use the compare editor to resolve the differences between the two files.



There are two parts to the compare editor's local toolbar. Move to the next or previous change using the right group of local toolbar buttons.



- Click the **Select Next Change** button . Observe how it selects the next difference.
- Click **Select Next Change button** a second time to go to the next change.
- Click the **Select Previous Change** button.

To merge changes from the left file to the right file and vice versa use the left group of local toolbar buttons. There are four types of merges that can be performed:

- Copy whole document from left to right.
- Copy whole document from right to left.
- Copy current change from left to right.
- Copy current change from right to left.

Typically the copy whole document actions are used when the entire file on either the left or right can just be replaced by the contents of the other file.

The Copy current change buttons allow a single change to be merged.

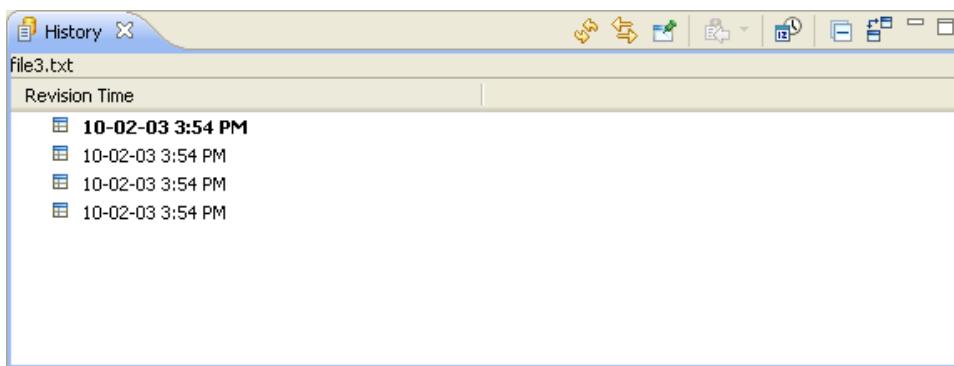
- Ensure that the second difference is selected:

-
- Click **Copy Current Change** from Right to Left . Observe that the selected text from the right file is copied to the left file.
 - Close the compare editor and choose **Yes** to save the changes. Alternatively, save the changes by choosing **File > Save** (Ctrl+S).

Local history

Every time an editable file is saved in the Workbench, the Workbench updates the local history of that file and logs the changes that have been made. The local history of a file can then be accessed and a previously saved copy of the file can be reverted to, as long as the desired state is recent enough in the save history.

- Create a new file named file3.txt.
- In the editor for file3.txt modify the file by adding the line "change1" and saving the file.
- Repeat this by entering a new line "change2" and saving it again.
- Add a third line "change3" and save it again.
- Right-click the file in a navigation view (e.g. the Project Explorer) and select **Team > Show Local History**.
- The History view opens and shows the history for the file.



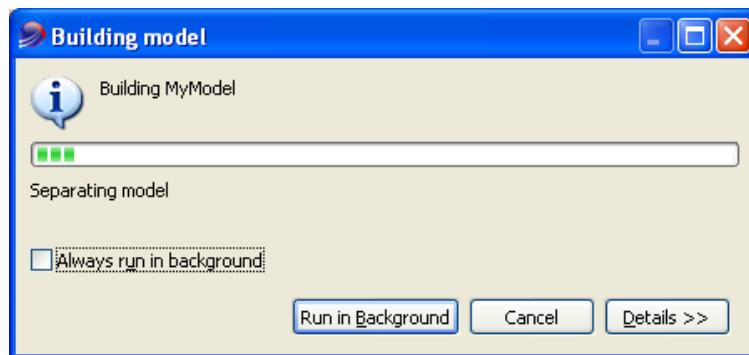
- The top entry in the view represents the current contents of the file. The next represents the previous contents and so on. Right-click on the previous entry and select **Compare Current with Local** to open a Compare editor that displays the differences between the Workbench file and the specific copy of the file selected in the local history.
- Right-click on the previous entry again and select **Get Contents**. This replaces the Workbench's copy of sampleFile.txt with the chosen local history item.
- Observe that the file3.txt editor now contains two lines.

Responsive UI

By default all RT-LAB operations run in the user interface thread. Employing the Responsive UI, which allows the threading of sequential code, will enable you to continue working elsewhere in RT-LAB. Without the Responsive UI support you would be locked out of performing any other actions when met with a slow operation.

While some operations automatically run in the background (such as build), in many cases a dialog will be displayed providing you with the option to run an operation in the background. For example, building a model can sometimes take more than a few minutes, during which time you may wish to continue to use other functions in RT-LAB.

While the model is being built, select **Run in Background** from the **Building Model** dialog and the Responsive UI will allow you to carry on with other tasks in RT-LAB.



For information on the status of the action and additional operations that are currently running, click Details.

The Details panel displays the status information of the operation at hand as well as any additional operations that may be running simultaneously.

The Progress Information dialog also indicates when one operation is being blocked by another.

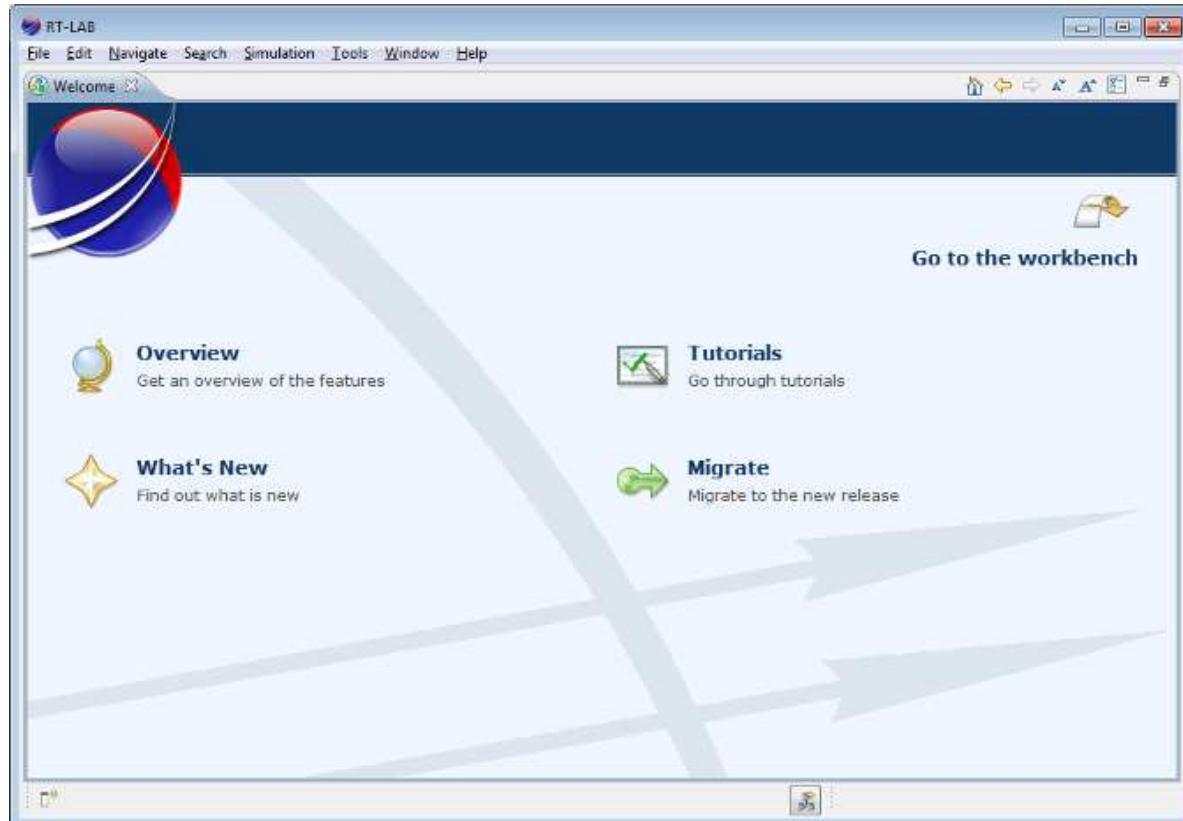
Exiting the Workbench

Each time the Workbench is exited, the Workbench is automatically saved, including all open perspectives and windows. The next time the Workbench is reopened, it will appear exactly as it was when it was last closed.

To exit the Workbench, select **File > Exit** from the menu bar or close the workbench with the window close button (x). When the latter option is used a prompt will ask if you really wish to exit the workbench.

Welcome

The welcome page is the first page you see when you first launch RT-LAB. Its purpose is to introduce you to the product. Welcome content typically includes an overview the product and its features, tutorials to guide you through some basic tasks, samples to get you started, etc.



Concepts

Contents:

- [**Welcome**](#)
- [**Workbench**](#)
- [**Perspectives**](#)
- [**Editors**](#)
- [**Views**](#)
- [**Workbench Menus**](#)
- [**Toolbars**](#)
- [**Wizards**](#)
- [**Preferences**](#)
- [**Cheat Sheets**](#)
- [**MetaController**](#)
- [**Help**](#)

Workbench

The term Workbench refers to the desktop development environment. The Workbench aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workspace resources.

Each Workbench window contains one or more perspectives. Perspectives contain views and editors and control what appears in certain menus and tool bars.

Related concepts

[Resources](#)

[Perspectives](#)

[Views](#)

[Editors](#)

Resources

Resources is a collective term for the projects, folders, and files that exist in the Workbench. The navigation views provide a hierarchical view of resources and allows you to open them for editing. Other tools may display and handle these resources differently.

There are three basic types of resources that exist in the Workbench:

- **Files** : Comparable to files as you see them in the file system.
- **Folders** : Comparable to directories on a file system. In the Workbench, folders are contained in projects or other folders. Folders can contain files and other folders.
- **Projects** : Contain folders and files. Projects are used for sharing, and resource organization. Like folders, projects map to directories in the file system. (When you create a project, you specify a location for it in the file system.)

A project is either open or closed. When a project is closed, it cannot be changed in the Workbench. The resources of a closed project will not appear in the Workbench, but the resources still reside on the local file system. Closed projects require less memory.

When a project is open, the structure of the project can be changed and you will see the contents.

Folders and files can be linked to locations in the file system outside of the project's location. These special folders and files are called linked resources.

Related concepts

- [**Workbench**](#)
- [**Project Explorer View**](#)
- [**Resource hierarchies**](#)
- [**Linked resources**](#)

Resource hierarchies

Resources are stored and displayed in the Workbench in hierarchies. Described below are the terms used when referring to resources that are stored and displayed in a hierarchical structure.

- **Root** : The top level of the Workbench contents (in the file system).
- **Parent resource** : Any resource that contains another resource. Only projects and folders can be parent resources.
- **Child resource** : Any resource that is contained within another resource. Only files and folders can be child resources.

Resource hierarchies are displayed in the Project Explorer view, which is one of the default views in the RT-LAB Edition perspective.

Related concepts

[Resources](#)
[Project Explorer view](#)

Linked resources

Linked resources are files and folders that are stored in locations in the file system outside of the project's location. These special resources can be used to add files and folders to your project that for some reason must be stored in a certain place outside of your project.

You can even use linked resources to overlap other resources in the workspace, so resources from one project can appear in another project. If you do want to have overlapping resources in your workspace, do so with caution. Keep in mind that this means changing a resource in one place will cause simultaneous changes in the duplicate resource. Deleting one duplicate resource will delete both!

Deleting a linked resource will not cause the corresponding resource in the file system to be deleted. However, deleting child resources of linked folders will cause them to be removed from the file system.

Related concepts

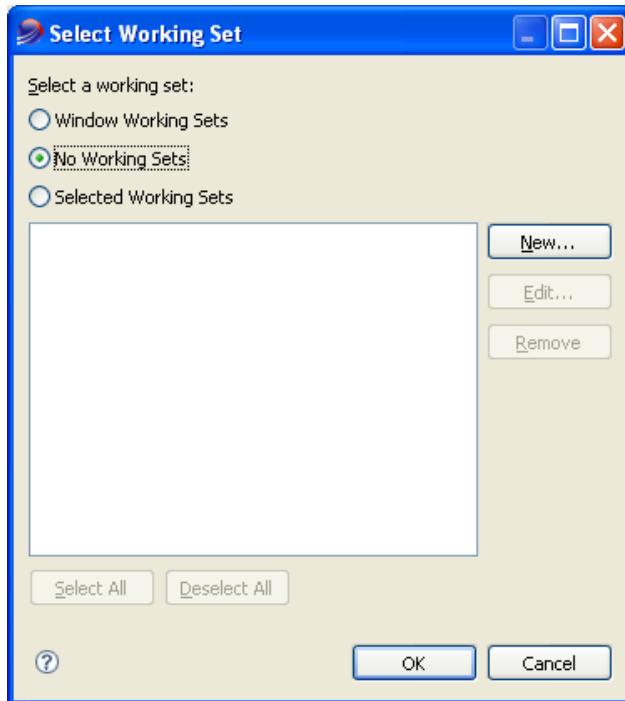
- [**Workbench**](#)
- [**Project Explorer view**](#)
- [**Resources**](#)
- [**Resource hierarchies**](#)

Working Sets

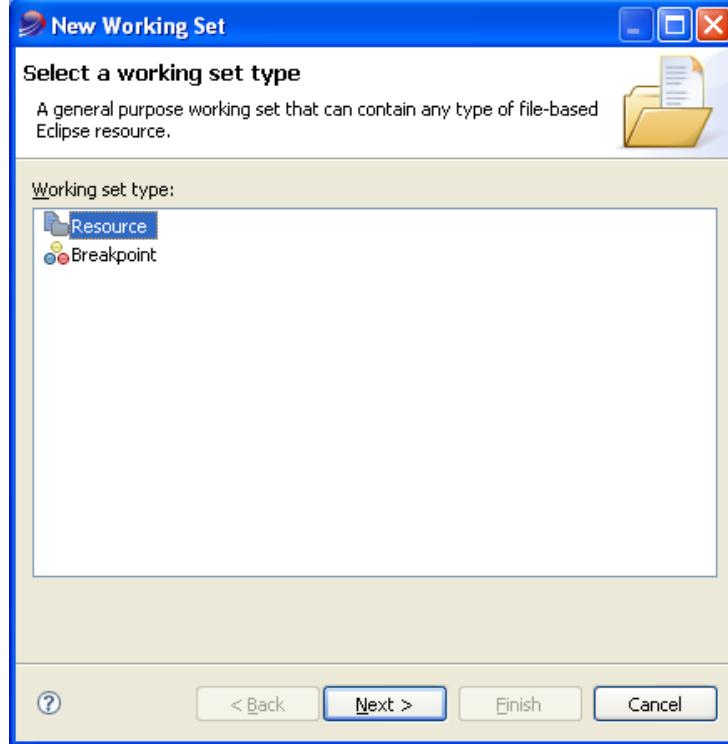
Working sets group elements for display in views or for operations on a set of elements.

The Project Explorer view uses working sets to restrict the set of resources that are displayed. If a working set is selected in the view, only resources, children of resources, and parents of resources contained in the working set are shown. When using the search facility, you can also use working sets to restrict the set of elements that are searched.

Different views provide different ways to specify a working set. The working Set of the Project Explorer view is specified in the **Working Sets** menu of the pull down menu of the Project Explorer view and is initially empty. Views that support working sets typically use the following working set selection dialog to manage existing working sets and to create new working sets:



When you create a new working set you can choose from different types of working sets. In the example below you can create a resource working set or a break point working set.



If you create a new resource working set you will be able to select the working set resources as shown below. The same wizard is used to edit an existing working set. Different types of working sets provide different kinds of working set editing wizards.

Note: Newly created resources are not automatically included in the active working set. They are implicitly included in a working set if they are children of an existing working set element. If you want to include other resources after you have created them you have to explicitly add them to the working set.

Related concepts

[Project Explorer view](#)

Local history

A local edit history of a file is maintained when you create or modify a file using the RT-LAB editors. Each time you edit and save the file, a copy is saved so that you can replace the current file with a previous edit or even restore a deleted file. You can also compare the contents of all the local edits. Each edit in the local history is uniquely represented by the date and time the file was saved.

Only files have local history; projects and folders do not.

Perspectives

Each Workbench window contains one or more perspectives. A perspective defines the initial set and layout of views in the Workbench window. Within the window, each perspective shares the same set of editors. Each perspective provides a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources. For example, the RT-LAB Edition perspective combines views that you would commonly use while editing RT-LAB model , while the Debug perspective contains the views that you would use while debugging Python scripts. As you work in the Workbench, you will probably switch perspectives frequently.

Perspectives control what appears in certain menus and toolbars. They define visible action sets, which you can change to customize a perspective. You can save a perspective that you build in this manner, making your own custom perspective that you can open again later.

Related concepts

[Workbench](#)

[Views](#)

[Editors](#)

Editors

Most perspectives in the Workbench are comprised of an editor area and one or more views.

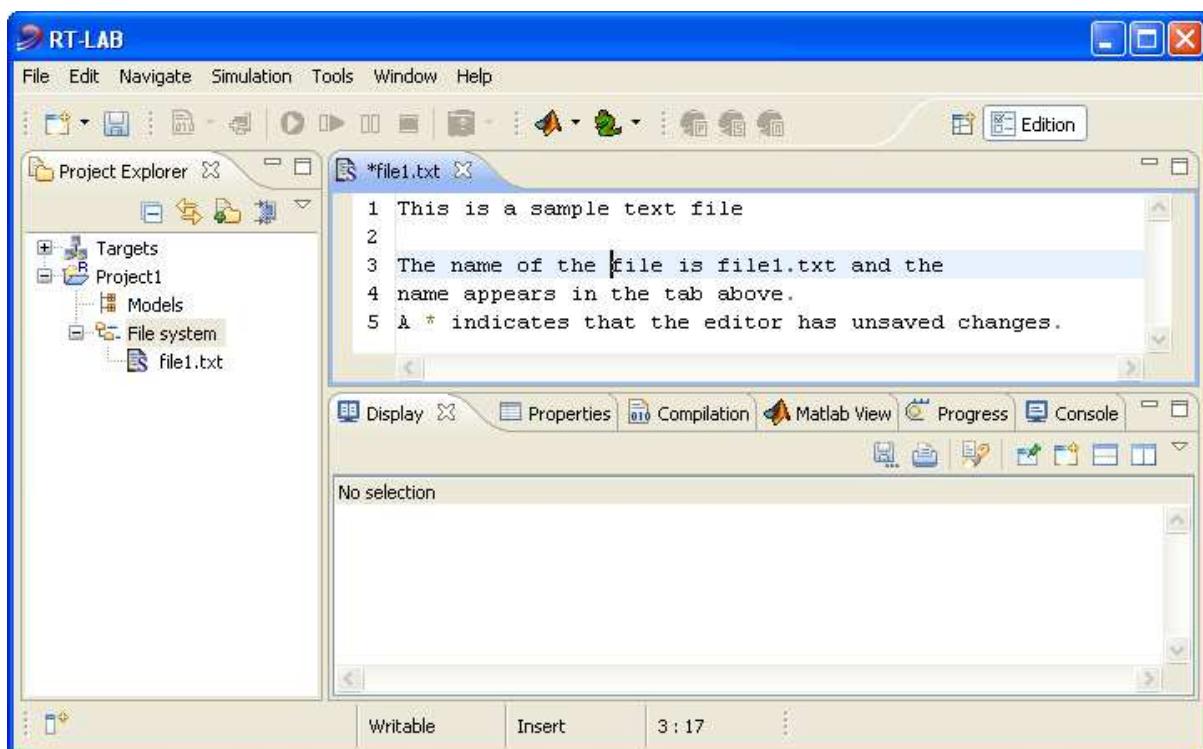
You can associate different editors with different types of files. For example, when you open a file for editing by double-clicking it in the Project Explorer view, the associated editor opens in the Workbench. If there is no associated editor for a resource, the Workbench attempts to launch an external editor outside the Workbench. (On Windows, the Workbench will first attempt to launch the editor in place as an OLE document. This type of editor is referred to as an embedded editor. For example, if you have a .doc file in the Workbench and Microsoft Word is registered as the editor for .doc files in your operating system, then opening the file will launch Word as an OLE document within the Workbench editor area. The Workbench menu bar and toolbar will be updated with options for Microsoft Word.)

Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for the Workbench window contain operations that are applicable to the active editor.

Tabs in the editor area indicate the names of resources that are currently open for editing. An asterisk (*) indicates that an editor has unsaved changes.

By default, editors are stacked in the editor area, but you can choose to tile them in order to view source files simultaneously.

Here is an example of a text editor in the Workbench:



The gray border at the left margin of the editor area may contain icons that flag errors, warnings, or problems detected by the system. Icons also appear if you have created bookmarks, added breakpoints for debugging, or recorded notes in the Tasks view. You can view details for any icons in the left margin of the editor by moving the mouse cursor over them.

Related concepts

[Workbench](#)

[External editors](#)

[**Model Editor**](#)
[**Target Editor**](#)
[**Project Explorer view**](#)

External editors

Sometimes you may need to use an external program to edit a file in the Workbench. This can occur, for example, when the Workbench has no editor for that file type.

The external program will be used as the default editor if that program is registered as the system default editor for that file type and no other editor is registered for that file type in the Workbench. For example, on most systems the default editor for JPEG files is an application for editing or viewing image files. If there is no other editor associated with .jpg or .jpeg files in the Workbench, then opening a JPEG file from the Workbench would cause the file to be opened externally, in the system default editor.

To open an external program that is not the default editor, you can use **Open With > Other...** from the context menu of a file. External programs can also be registered in RT-LAB as the associated editor for a given file type. Use the **General > Editors > File Associations** preference page to register editors.

Related concepts

[Workbench](#)
[Editors](#)

Model Editor

RT-LAB provides a simple form-based multi-page model editor that manages all model properties. To open a user-friendly model editor into the current perspective, double-click on your model in the Project Explorer. Note that all model properties are also available in the Properties View when selecting a model.

An RT-LAB model is characterized by a Simulink model and a set of properties. These properties are not necessarily linked to a Simulink model. Sometimes these properties are related to an RT-LAB simulator.

These properties are named Model Properties and are defined for each new RT-LAB model.

Each new model inherits its default property values from the RT-LAB preferences (defined in the Preference Pages of RT-LAB). These properties are individually editable for each RT-LAB model through the Model Editor or Model Properties View. You can configure several models at the same time by using the Model [Properties View](#).

These properties are organized by categories. Each category is shown on different page of the model editor. One or more categories can be displayed on the same page when related.

For details on the individual editor pages, refer to the following documents :

- [Model Overview Page](#)
- [Development Page](#) (including compiler and linker properties)
- [Execution Page](#) (including real-time and performance properties)
- [Environment Variables Page](#)
- [Files Page](#)
- [Assignation Page](#) (including subsystem properties and utilities)
- [Diagnostic Page](#) (including monitoring and debugging properties)
- [Hardware Page](#) (including monitoring and performance properties)
- [Simulation Tools Page](#) (including Matlab/Simulink properties)

Each modification of a model property covers only the edited model and is saved only for the edited model. A modification made in the "Preference" menu will not change edited model but will be effective for each new RT-LAB model created after the modification.

Model Overview Page

The Overview page contains three main sections that define important model properties:

- General Information
- Preparing and compiling
- Executing

General Information

General Information

This section describes general information about this model.

Name:	rtdemo1
Path:	C:/Git/RT-LAB/build/release/win32/workspace/rtdemo1/Simulink/rtdemo1.mdl
State:	Loadable
Description:	RTDemo1 The rtdemo1 model demonstrates ... What is a PID controller? PID stands for Proportional-Integral-Derivative. This is a type of feedback controller whose output, a control variable (CV), is generally based on the error between some user-defined set point (SP) and some measured process variable (PV). Each element of the PID controller refers to a particular action taken on the error: Proportional: Error multiplied by a gain, Kp. This is an adjustable amplifier. In many systems Kp is responsible for process stability: too low and the PV can drift away; too high and the PV can oscillate. Integral: The integral of error multiplied by a gain, Ki. In many systems Ki is responsible for driving error to zero, but to set Ki too

This section describes general information about the edited model.

Name is the name of the edited model.

Path is the path where the model is located on the file system.

State is the current state of your model. (See "[OP_MODEL_STATE](#)")

Description is the description of the model.

Note: If the model is not loadable, an error is displayed in the State field:

State:  Not loadable <Not compiled for Redhat>

Figure 1:State: not loadable

Preparing and compiling

Preparing and Compiling

To prepare and compile the model:

-  [Edit](#) the diagram of the model.
-  [Set](#) the development properties.
-  [Compile](#) the model.
-  Consult result in the [Compilation View](#)
-  Check errors in the [Problem View](#).
-  [Assign](#) targets to subsystems.

This section provides some shortcuts relative to the preparation and to the compilation of the edited model.

Edit the diagram model: Opens the model with the current Matlab version.

Set development properties: Opens the [Development Page](#) of this Model Editor.

Build the model: Builds the model with RT-LAB.

Consult result in the Compilaiton view: Opens the [Compilation View](#).

Check errors in the Problem View: Opens the [Problems view](#).

Assign targets to subsystems: Opens the [Assigantion Page](#) of this Model Editor.

Executing

Executing

To execute the model:

[Set](#) the execution properties.

[Load](#), [Execute](#), [Pause](#), [Reset](#) the model,

Check execution results in the [Display View](#).

Check execution warnings and errors in the [Problem View](#).

This section provides some shortcuts relative to the model execution.

Set the execution properties: Opens the [Execution Page](#) of this Model Editor.

Load: Loads the executable code onto the target nodes (only available when the model is loadable).

Execute: Starts the simulator and the execution of the distributed model's simulation on the network of target. (only available when the model is paused).

Pause: Puts the system in a waiting state, stopping calculation without resetting the nodes. To continue execution, click on Execute. (only available when the model is running).

Reset: Terminates the execution of the executable(s) on the target nodes and causes a complete stop of the simulator. (only available when the model is running or paused).

Check execution results in the Display View: Opens the [Display View](#).

Check execution warnings and errors in the Problem View: Opens the [Problems view](#).

Development Page

This page contains two main sections that define development settings for the edited model : **target platform** and **compiler / linker settings**.

The default property values for a new model come from the RT-LAB Preference pages. If a property is changed from the RT-LAB preference pages, each new model created will inherit from these properties. After that, if you change a model property, only the edited model will be affected.

Target Platform section

Compiler and linker settings

Customize the compiler and linker settings.

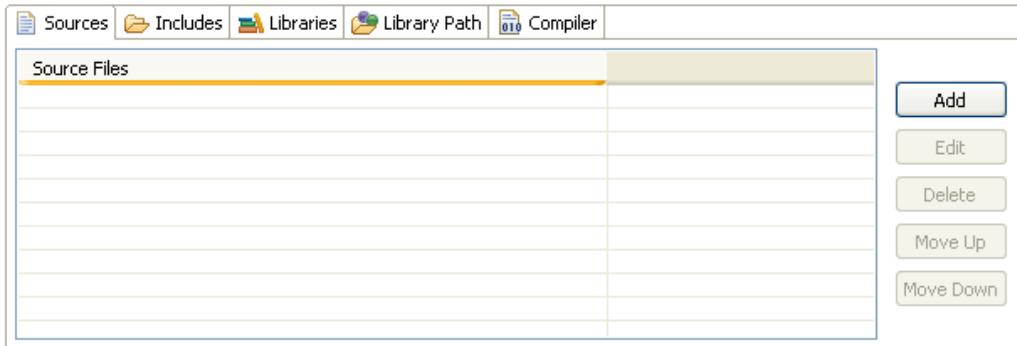
Target platform:

Target platform: Allows you to choose the target platform (or the operating system) on which to run the simulation (QNX6, Redhat, Redhawk, Windows XP/Vista/7 system).

Compiler / Linker settings section

This section is composed of five sub-sections : sources, includes, libraries, library path, and compiler.

Sources



Sources: Names of source files to be compiled in addition to the code generated by RTW. Used to incorporate user-written code into the model executable (e.g. file1.c file2.c file3.c).

Add: Adds a source file to the list of source files.

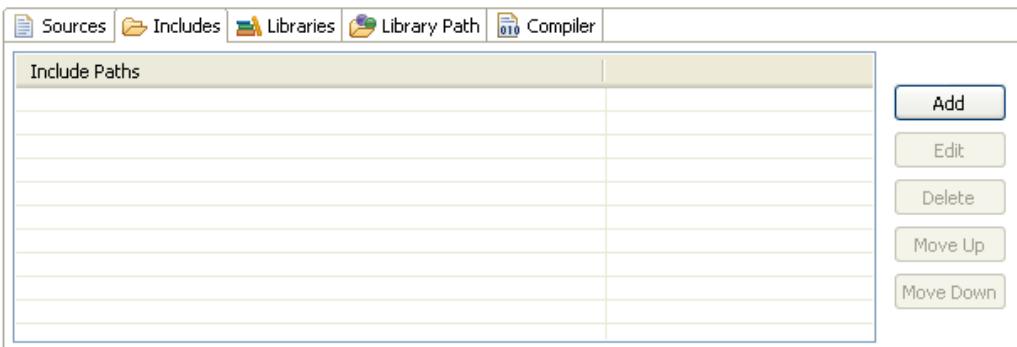
Edit: Edits the path and name of the selected source file.

Delete: Removes a source file from the list.

Move up: Changes the order of the source files; moves up the selected source file in the list.

Move down: Changes the order of the source files; moves down the selected source file in the list.

Includes



Includes: Target path(s) where user include files can be found. Used to search for include files required to compile user-written code, when the include files do not reside in already-searched paths.

Add: Adds an include path to the list.

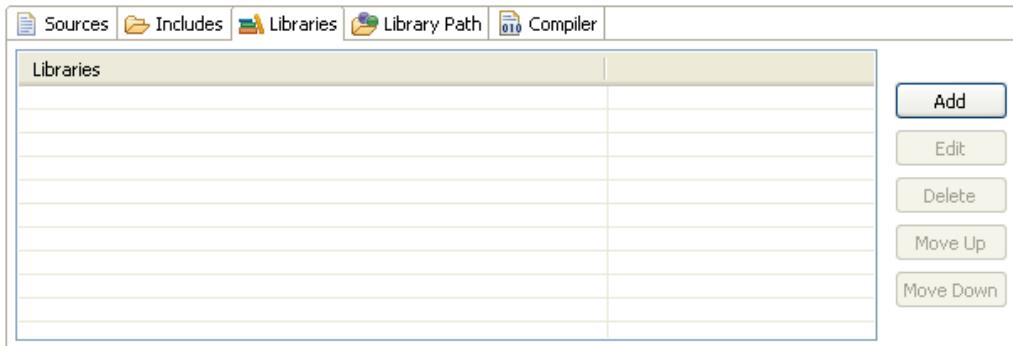
Edit: Edits the selected include path.

Delete: Removes the selected include path from the list.

Move up: Changes the order of the include paths; moves up the selected include path in the list.

Move down: Changes the order of the include paths; moves down the selected include path in the list.

Libraries



Libraries: Names of user-specified libraries to be used when incorporating user-written code, in addition to the standard system libraries (Matlab and RT-LAB).

Add: Adds the path of a library to the list.

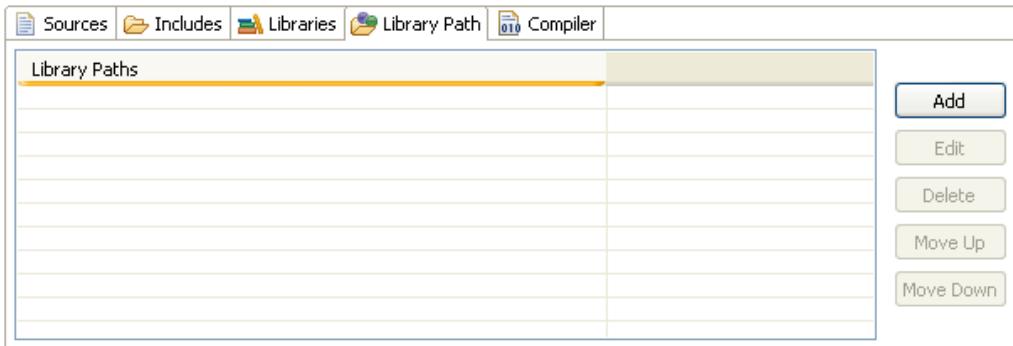
Edit: Edits the path of the selected library.

Delete: Removes the selected library from the list.

Move up: Changes the order of the libraries; moves up the selected library in the list.

Move down: Changes the order of the libraries; moves down the selected library in the list.

Library path



Library path: Target path(s) where user libraries can be found. Used to search for libraries other than the MATLAB and RT-LAB libraries when incorporating user-written code.

Add: Adds a library path to the list.

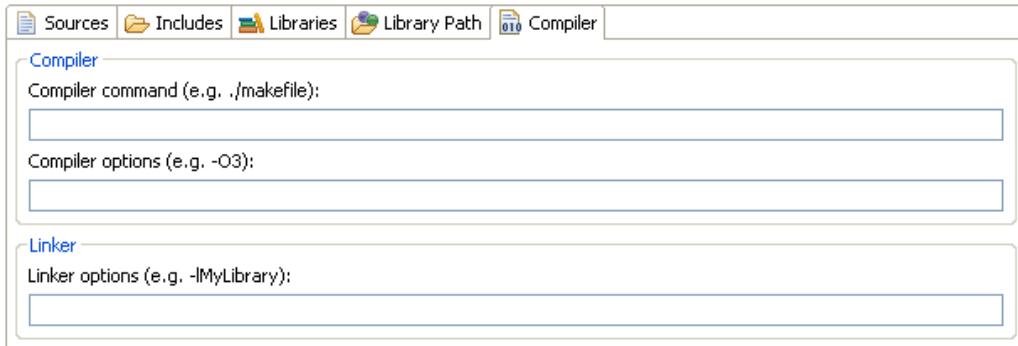
Edit: Edits the selected library path.

Delete: Removes the selected library path from the list.

Move up: Changes the order of the libraries path; moves up the selected library path.

Move down: Changes the order of the libraries path; moves down the selected library path.

Compiler



Compiler Command: User command to be called on the target during the compilation process.

Compiler Options: Options used by the target compiler during the compilation process. Refer to the target compiler documentation for specific information on the possible compiler options (e.g. -xxx).

Linker Options: Options used by the target linker during the linkage process. Refer to the target linker documentation for specific information on the possible linker options (e.g. -xxx).

Execution Page

This page contains two main sections that define execution settings for the model : **Real-Time Properties** and **Performance Properties**.

The default property values for a new model come from the RT-LAB Preference pages. If a property is changed from the RT-LAB preference pages, each new model created will inherit from these properties. After that, if you change a model property, only the edited model will be affected.

Real-Time Properties

Real-Time Properties

Set the real-time properties of the model.

Target platform:	Windows XP/Vista
Real-time simulation mode:	Software synchronized
Real-time communication link type:	UDP/IP
Time Factor:	1.0
Stop Time [s]:	Infinity
Pause Time [s]:	Infinity
Advanced	
<input type="checkbox"/> Use restricted mode	
Restricted clock time step [us]:	1

Target platform: Allows the user to choose the target platform on which to run the simulation (QNX6, Redhat, Redhawk, Windows XP/Vista/7 system).

Real-Time simulation mode: Enables the user to define how the model's simulation can be executed on a QNX6, RedHat, or Windows XP/Vista/7 system.

- Simulation: in this mode, the model is not synchronized; the model starts a new computation step as soon as the previous one is completed (the model runs as fast as possible).

- Simulation with low priority: this simulation mode can be used to run a simulation in the background while working with other applications on the same computer.

Note: Please note that the Simulation with low priority mode is only available when working with a Windows target.

- Software Synchronized: in this mode, real-time synchronization of the entire simulation is achieved by the OS, using the CPU clock as a reference. Depending on the resolution available on the OS, some sampling times may not be obtained using this mode (e.g. on QNX6.1 the smallest sampling time available is 500 µs).
- Hardware Synchronized: in this mode, an I/O board clock is used to synchronize the entire simulation.

Real-Time communication link type: Sets the target cluster's communication medium.

Here are the available choices for each target platform.

- XP/Vista/7 system: UDP/IP.
- QNX 6.x: OHCI, UDP/IP.
- RedHat: OHCI, UDP/IP, SCI, INFINIBAND.

Time Factor: This value, when multiplied by the model's basic calculation step (or fixed step size), supplies the final calculation step for the system. The time factor can only be changed when the simulation execution is paused and is only available in Synchronized (hardware/software) mode.

Note: This parameter enables you to change the I/O acquisition speed. Because the model always uses the basic calculation step, changing the time factor for the I/O may give results which differ from the offline simulation. This parameter should be used only to determine the simulation's minimum calculation step. Once this step is determined, this parameter must be brought back to a value of 1. The model's basic calculation step must be updated accordingly by setting the correct value in the Simulink model and by recompiling it.

Stop Time [s]: This value (in seconds) specifies when the model will stop. If value is "Infinity", the model will execute forever. You don't need to recompile the model to apply a new value.

Pause Time [s]: This value (in seconds) specifies when the model will pause. If value is "Infinity", the model will execute until a reset is done except if a stop time is specified.

Use restricted mode (For advanced users only): This setting can be used only when the model's current target platform is QNX 6.x. When this platform is selected, a given model can be loaded either in 'Free Clock' mode or in 'Restricted Clock' mode.

- In Free-Clock mode, the user can load one non-Software Synchronized model on a given target node and is free to use any time-step value (within the boundaries of the QNX 6.x restrictions). If the user tries to load another non-Software Synchronized model on the same target node, it will be stopped with a relevant error message.
- In Restricted-Clock mode, the user can load a series of non-Software Synchronized models on a given node. In this mode, each model time-step must be consistent (i.e. a multiple of) with a specific base Clock Period that should be specified by the user. Each time an additional model is being loaded on this specific target, its time-step will be checked against the current base Clock Period to validate the load operation.

Restricted clock time step [us] (For advanced user only): Defines the Clock Period in microseconds specified by the user for the Use Restricted mode option. (Only available when the Use Restricted mode option is enabled).

Performance Properties

Performance Properties

Set the performance properties of the model.

Enable detection of overruns

Action to perform on overruns: Continue

Perform action after N overruns: 10

Number of steps without overruns: 10

Enable detection of overruns: (Only applies to models ran in hardware/software synchronized modes.) When this parameter is not checked, RT-LAB does not detect overruns. When this parameters is checked, RT-LAB detects overruns and performs the action specified in the "Action to perform on overruns" field. This property has no effect in simulation mode.

Action to perform on overruns: Type of action to perform when overruns are detected.

Here are the available types:

- Continue: No action is perform ; Number of overruns will simply be displayed during the next pause/reset of model.
- Reset: A reset of the model is performed when the number of overruns reaches the value of the "Perform action after N overruns" field.
- Pause: A pause of the model is executed when the number of overruns reaches the value of the "Perform action after N overruns" field.

Perform action after N overruns: Performs the action specified in the "Action to perform on overruns" field when the number of overruns detected by RT-LAB reaches this number. (Only applies when "Action to perform on overruns" is set to Reset or Pause.)

Number of steps without overruns: Prevents RT-LAB to detect overruns during the first steps of simulation.

Environment Variables Page

This page gives a list of environment variables defined for the edited model.

User environment variables are used to enable some settings for the simulation without having to recompile the model. Most of these settings are used to debug models or as workarounds for specific target situations.

The default property values for a new model come from the RT-LAB Preference pages. If a property is changed from the RT-LAB preference pages, each new model created will inherit from these properties. After that, if you change a model property, only the edited model will be affected.

Environment Variables

Set the environment variables of your model.

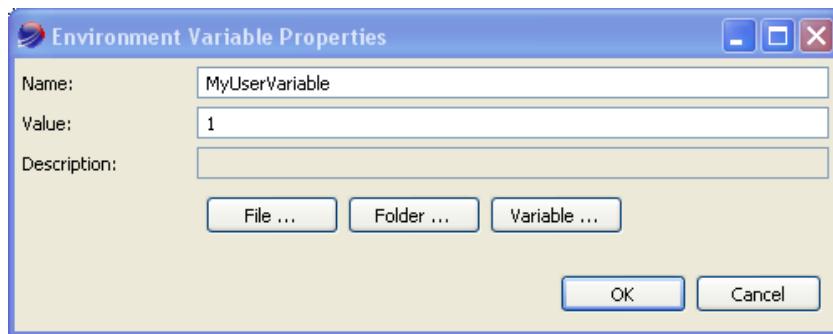
Name	Value	Description	
ARTEMIS_VERSION	v5.1.2		Add...
			Select...
			Edit...
			Delete
			Unset

Name: Name of the user variable.

Value: Value of the user variable.

Description: Description of the user variable.

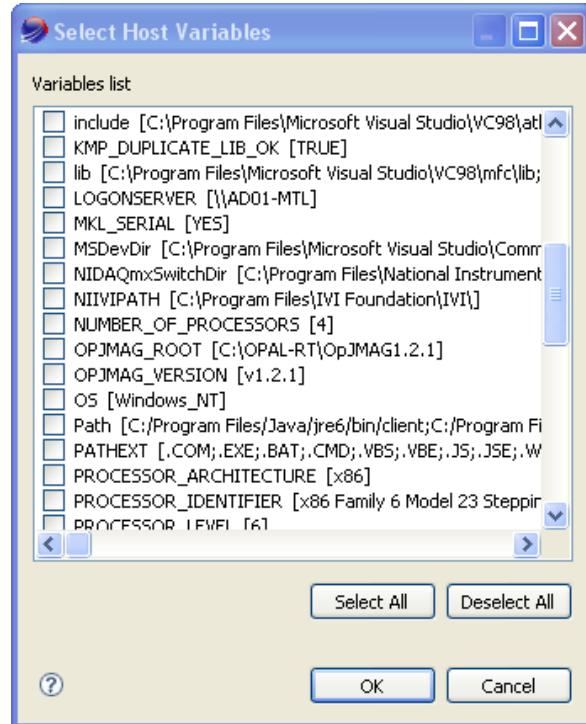
Add a new user variable



To add a new user variable :

1. Click on the Add button.
 2. Fill the Name field which is the name of your new variable.
 3. Fill the Value field which is the value of your new variable.
 - Type a value
 - Select a path to a file or a folder by clicking on the “File...” or “Folder...” button
 - Get the value of an host environment variable by clicking on the “Variable...” button
 4. Click on the OK button.

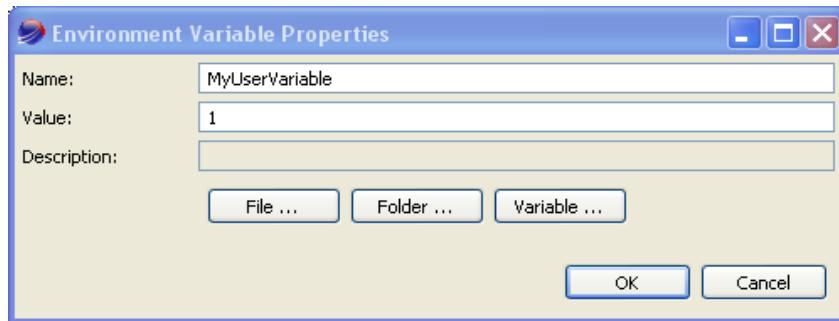
Select an environment variable



Sometime, it can be useful to import an existing variable from the host system to the target nodes. To select an existing environment variable on your host system:

1. Click on the Select button.
2. Select all the desired environment variables that you want to import. If you want to select/deselect all environment variables, you CAN click on the "Select All" and the "Deselect All" buttons respectively.
3. Click on the OK button.

Edit an user variable



To modify an user variable :

1. Select an user variable from the list to modify.
2. Click on the Edit button.
3. Change the desired field : Name, Value or Description field.
For the Value field, you could choose between these options:
 - Fill a value manually.
 - Select a path to a file or a folder by clicking on "File..." or "Folder..." button

- Get the value of an host environment variable by clicking on "Variable..." button
 - Click on OK button.

Delete an user variable

To remove an user variable:

- Select the user variable from the list to delete.
- Click on the Delete button.

Set / Unset an user variable

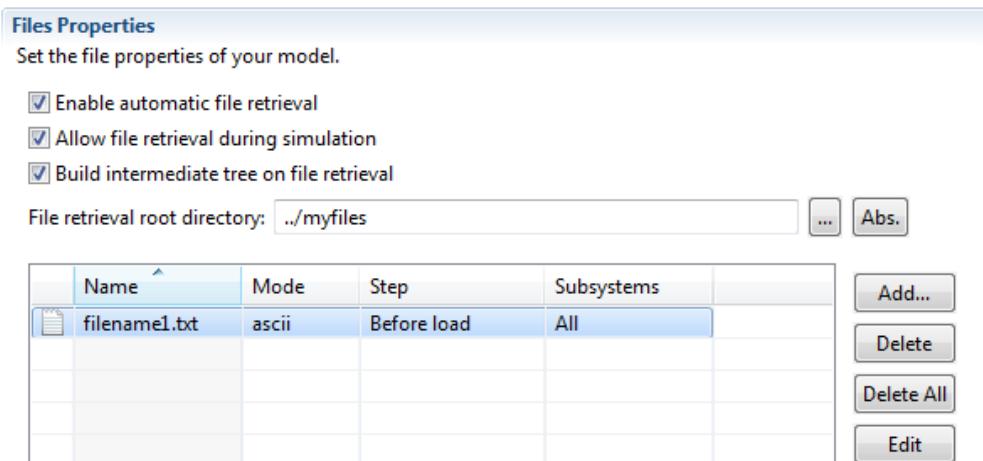
You can also Set or Unset an user variable to activate/deactivate variables on the target nodes. Unset variables are unavailable on the target.

- Select the user variable you want to Set or Unset.
- Click on the Set/Unset button.

Files Page

This page shows the files properties for the edited model.

The default property values for a new model come from the RT-LAB Preference pages. If a property is changed from the RT-LAB preference pages, each new model created will inherit from these properties. After that, if you change a model property, only the edited model will be affected.



Enable automatic file retrieval: When this option is enabled, RT-LAB retrieves files generated by the model simulation on the target nodes and transfers them to the host computer.

Enable file retrieval during simulation: If this option is enabled, RT-LAB retrieves the files generated by the model simulation as soon as they are created on the target. If this option is disabled, RT-LAB will retrieve those files after the model resets.

Build intermediate tree on file retrieval: Specifies if intermediate folders are generated or not in the retrieve directory.

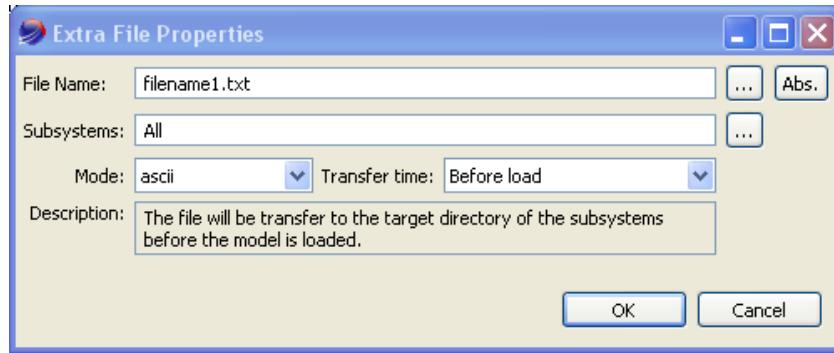
File retrieval root directory: Directory where the files retrieved by RT-LAB after the model execution will be copied.

List of extra-files: This list specifies one or more files to be copied and transferred to (or from) the target environment.

Here are the properties for each extra file.

- Name: Extra file path and name.
- Mode: File transfer type (ascii or binary).
- Transfer time: Time when file will be transferred. Here are the different transfert times
 - Before compilation: File will be transferred before compilation process.
 - After compilation: File will be transferred after compilation process.
 - Before load: File will be transferred before the load of model.
 - After reset: File will be transferred after the reset of model.
- Subsystem: Subsystem directory where files should be transferred. Type All if you want to transfert file for each subsystem. Note that a unique folder exists for each subsystem on the target nodes and on the host computer.

Add an extra-file to list



To add a new extra-file :

1. Click on the Add button.
2. Fill the File Name field or select a file from the file system by clicking on the "...button. You can select an absolute path by clicking on the "Abs"button.
3. Select the subsystem directory where files should be transferred. If you want to transfer the files on All subsystem directory, type "All" on field Subsystems.
4. Select the transfert mode of your file.
5. Select the transfert time of your file.
6. Click on the OK button to add the file.

Delete an extra-file from list

To remove an extra-file from list :

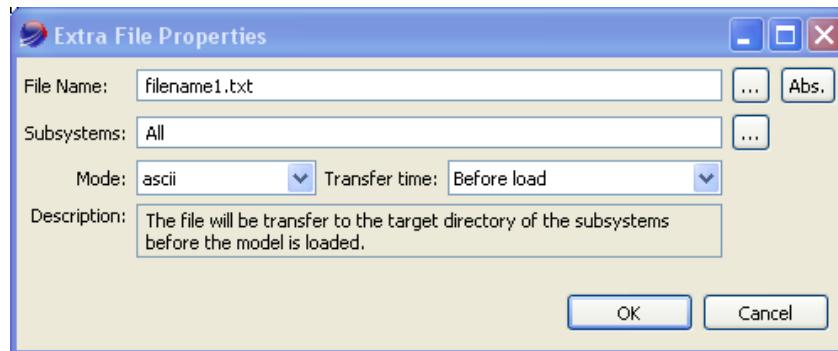
1. Select the extra-file from the list.
2. Click on the Delete button.

Delete all extra-files from list

To remove all extra-files from list :

1. Click on the Delete All button.

Edit an extra-file



To edit an extra file in list :

1. Select an extra-file.
2. Modify the field(s) you want to change.
3. Click on the OK button to add the file.

Assignation Page

This page gives the subsystems properties for the edited model. The subsystems page contains two main sections that define the subsystems properties:

- Allows the user to assign the computation subsystems to the different target nodes
- Clean a target or to embed a subsystem

Assignations

This section provides a table that allows the user to set all the properties for subsystems of a real-time model. The editor allows the user to directly edit the properties in the table or to edit the selected subsystems in the field below.

Subsystem settings

Assignations

Set the properties of the subsystems and assign them to physical nodes.

Subsystems

Select subsystems to edit their properties:

Name	Assigned node	Platform	XHP	Debug	Cores
sm_controller	193.2x4	Redhat	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	1
ss_plant	193.2x4	Redhat	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	1

No subsystem selected

Edit settings for selected subsystems:

Choose a physical node:

Run in XHP mode

Advanced

Run in debug mode

The user can set properties for each subsystem directly in the subsystem table : target assignation, XHP mode, debug mode and core assignation.

Subsystems

Select subsystems to edit their properties:

Name	Assigned node	Platform	XHP	Debug	Cores
sm_controller	193.2x4	Redhat	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	1
ss_plant	193.2x4	Redhat	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	1

Name: Model's subsystem names.

Assigned Node: Select the target node where the subsystem will be executed from the list of available targets. By default, the Assigned node is set to the development node. See Preferences->RT-LAB->Simulation Tools->RT-LAB section.

Platform: Type of platform of the selected target. (QNX 6.X, Redhat, Windows).

XHP: If checked, the subsystem will be executed in eXtreme High Performance (XHP) mode (available only on QNX 6 and RedHat Linux).

Debug: Check the subsystems you want to debug. (See section Debugging on page 30.)

CPU: (For advanced users only). CPU cores allocated (and optionally specified) for the subsystem and its worker threads.

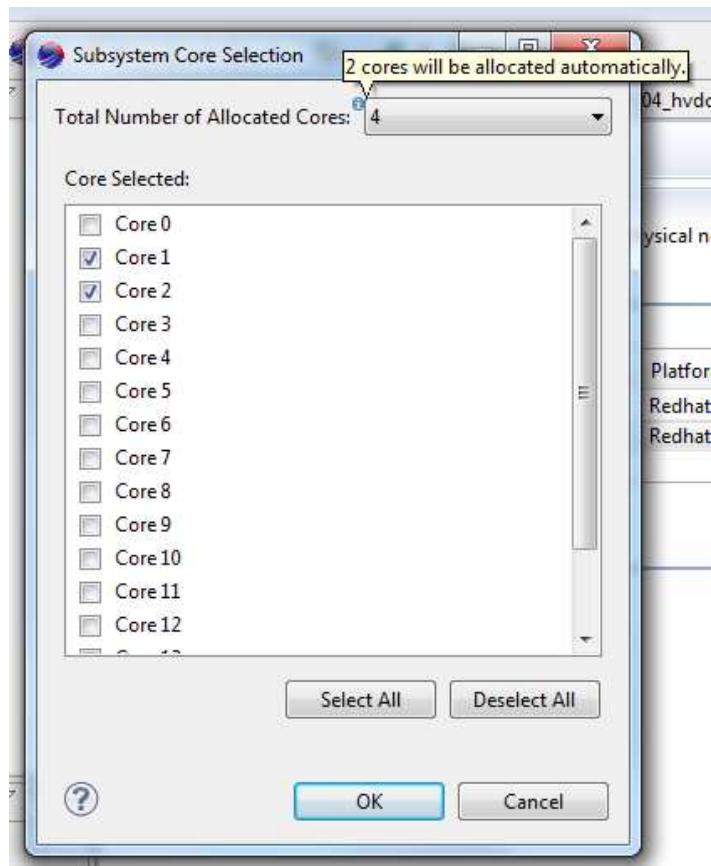
The system will allocate cores and will assign subsystems using the following set of rules:

- Single core systems do not support XHP mode.
- Assignment starts at CPU #1 on multiple core systems; CPU #0 is reserved.
- Specified cores are allocated first.
- Unspecified cores of XHP subsystems are allocated before unspecified cores of other subsystems.

- XHP subsystems cannot share cores with any other subsystem.
- Unspecified workers threads are allocated last using a fair distribution until no core are available.

Note that a system with N CPU has a maximum of N - 1 subsystems in XHP.

When the "Advanced" section is activated, select the "... in the "Cores" column to display the core selection dialog. Then, select the number of cores and optionally, select specific cores.



Multiple Assignment

When selecting multiple subsystems at the same time in the table, the user can set the properties for this group below the table: target assignation, XHP mode and debug mode. This can also be used when only one subsystem is selected in the table.

1 subsystem selected : ss_plant

Edit settings for selected subsystems:

Choose a physical node: **193.2x4**

Run in XHP mode

Advanced

Run in debug mode

Select one or N subsystem(s) in the Subsystem table and set the properties :

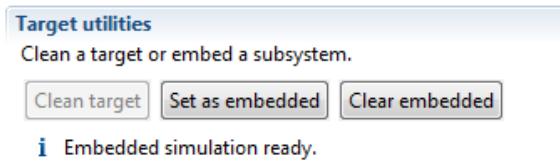
Choose a physical node: Select the target node where the subsystem will be executed from the list of available targets. By default, the Assigned node is set to your development node. See Preferences->RT-LAB->Simulation Tools->RT-LAB section.

Run in XHP mode: If checked, the subsystem will be executed in eXtreme High Performance (XHP) mode (available only on QNX 6 and RedHat Linux).

Run in Debug mode (For advanced users only): Check the subsystems you want to debug. (See section Debugging on page 30.)

Target Utilities

The target utilities section allows the user to clean a target or embed a subsystem.



Clean target: Cleans the model directory on the target node. All files will be deleted.

Set as embedded: Sets the current model's simulation as an embedded simulation on the target. It allows the target to automatically load and execute a simulation at power up. See [Embedding Simulation](#) for more details.

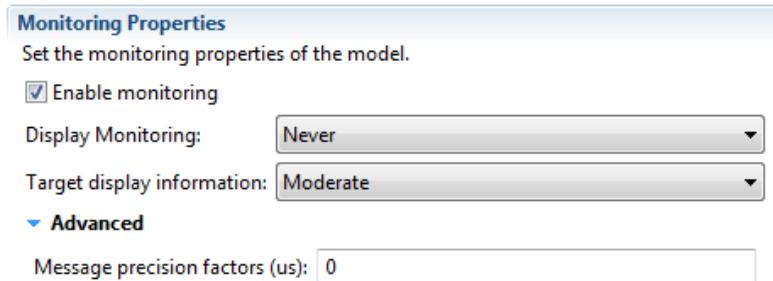
Clear embedded: Clear an embedded simulation from the assigned target. See [Embedding Simulation](#) for more detail.

Diagnostic Page

The diagnostic page contains two main sections that define the diagnostic properties : **Monitoring properties** and **Debugging properties**.

The default property values for a new model come from the RT-LAB Preference pages. If a property is changed from the RT-LAB preference pages, each new model created will inherit from these properties. After that, if you change a model property, only the edited model will be affected.

Monitoring Properties



Enable monitoring: Enables or disables the monitoring during the execution of the model.

Display Monitoring: Specifies when Monitoring results is to be displayed in the Display View (NEVER, AT_PAUSE, AT_RESET, or BOTH)

Target display information: Represents the amount of information that is displayed in the display window during load and execution. Setting the value from MINIMAL to EXHAUSTIVE allows the user to

retrieve more information about communication and I/Os initialization for example. This can be used to debug unexpected model behaviors.

Note: Please note that Exhaustive mode must only be used for debug purposes as it can influence the real-time performances.

Message precision factor [us] (For advanced users only): Defines the precision of the timer reporting the time required to perform one calculation step in microseconds. The default value is 0 (no printout). Value=1 prints the step size of the model every 1,000,000 steps with a precision of 1 μ s. Value=100 prints the model's step size every 10,000 steps with a precision of 100 μ s and so on.

Warning: This option may disturb the real-time simulation performances.

Debugging Properties

Debugging Properties

Set the debugging properties of the model.

Enable extended timeout
 Enable watchdog
 Watchdog timeout [ms]:
 Compile model in debug.

Enable extended timeout: All timeouts are extended when this option is enabled. It is typically used when debugging the model. Default value of 300 seconds.

Enable watchdog: Enables or disables the Watchdog functionality. The Watchdog ensures that all nodes are running normally. There are as many watchdogs as there are computation subsystems in the model. At regular user-defined time intervals, the watchdog verifies that the model is still running and that processes are still active. If an error occurs, the watchdog stops the simulation. This option is not available on Windows target.

Watchdog timeout [ms]: The regular user-defined time interval the watchdog verifies the model is still running. 5000 ms by default.

Compile model in debug: When this option is enabled, the compiler's optimization options are removed and replaced by debug options during model compilation. This option is useful when debugging a model.

Hardware Page

The Hardware page contains two main sections that define the hardware properties : **Monitoring properties** and **Performance properties**.

The default property values for a new model come from the RT-LAB Preference pages. If a property is changed from the RT-LAB preference pages, each new model created will inherit from these properties. After that, if you change a model property, only the edited model will be affected.

Monitoring Properties

Monitoring Properties

Set the monitoring properties of the hardware.

Reset model on IO missing
 Abort compilation on bitstream missing

Reset model on IO missing: When this option is checked and the model is loaded, RT-LAB will stop loading and will generate an error, if some I/O boards required by the model are not detected on the

target(s). RT-LAB will generate only a warning if this setting is unchecked. Note that only a few I/O boards (mainly the Opal-RT TestDrive boards) support this option. Most I/O blocks force a reset of the model if the boards are not detected at load time.

Abort compilation on bitstream missing: When this option is enabled, if some bitstreams required by a model are not present in the model directory, RT-LAB will stop the compilation and generate an error. RT-LAB will generate only a warning if this setting is disabled.

Performance Properties

Performance Properties
Set the performance properties of the hardware.
 Enable cacheable DMA memory access

Enable cacheable DMA memory access: When this option is checked, RT-LAB will enable the cache of the memory areas used by DMA buffers yielding to faster transfer rates and increased model performances. However, this option should be used with caution since it can affect proper reading of I/O data using DMA buffers. This option is available on QNX6 only.

Simulation Tools Page

The Simulation Tools Properties page contains three main sections that define common properties related to external simulation tools : **Matlab and Simulink properties**, **Simulink Console** and **Real-Time Workshop properties**.

The default property values for a new model come from the RT-LAB Preference pages. If a property is changed from the RT-LAB preference pages, each new model created will inherit from these properties. After that, if you change a model property, only the edited model will be affected.

Matlab and Simulink Properties

Matlab and Simulink Properties
Set the Matlab and Simulink properties of the model.

Model fixed step size [s]:

Matlab command before opening model (Example myVar=1):

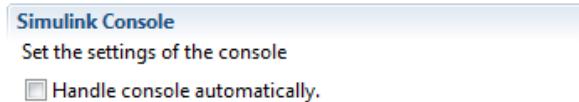
Matlab command after opening model (Example myVar=1):

Model fixed step size [s]: Displays the basic calculation step of the model as specified in the Simulation>Configuration Parameters>Solver menu of the Simulink model file. This property is read-only.

Matlab command before opening model: Matlab command to execute before opening the model. For example. MyInitFile.m, where MyInitFile includes commands that initialize model variables.

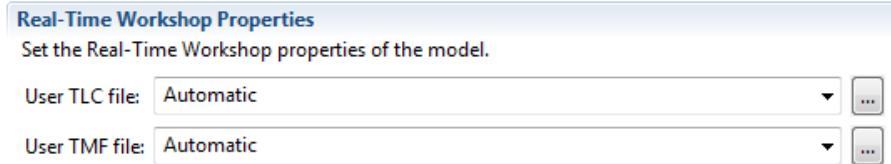
Matlab command after opening model: Matlab command to execute after opening the model. For example, MyInitFile.m, where MyInitFile includes commands that initialize model variables.

Simulink Console



Handle console automatically: Enables the console to be automatically opened and closed, when a model is Loaded or Reset. The console starts automatically at the first model's execution.

Real-Time Workshop Properties



User TLC file: Specifies which tlc file to use during the compilation. When Automatic is selected, the tlc file corresponding to MATLAB version is automatically selected. This option is available for SIMULINK models only.

User TMF file: Specifies which template makefile to use during the compilation. When Automatic is selected, the template makefile corresponding to MATLAB version is automatically selected. This option is available for SIMULINK models only.

Target Editor

RT-LAB provides a simple form-based multi-page target editor that manages all target properties. To open a user-friendly target editor into the current perspective, double-click on your target in the Project Explorer. Note that target properties are also available in the [Properties View](#) when selecting a target.

These properties are organized by categories. Categories are shown on different pages of the target editor. One or more categories can be displayed on the same page when related.

For details on the individual editor pages, refer to the following documents :

- [Target Overview Page](#)
- [Diagnostic Page](#)
- [Simulation Settings Page](#)
- [Software Page](#)

Target Overview Page

The Overview page contains two main sections that define important target properties and allow common actions:

- General Information
- Operations

General Information section

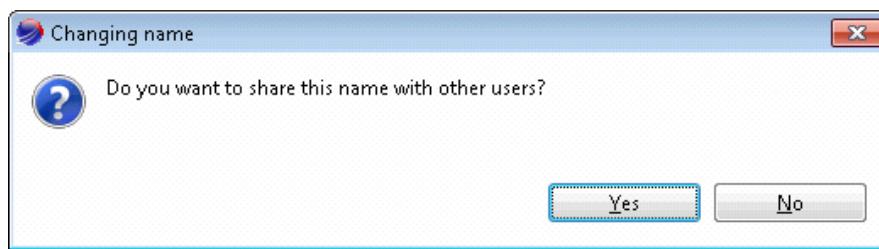
General Information

This section describes general information about this target.

Name:	MandaBox
IP address:	192.168.0.231
State:	Up

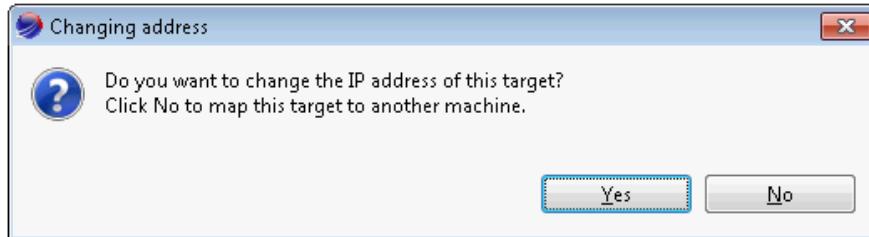
This section describes general information about the edited target.

Name is the name of the target. When you modify this name, you have the option to share this name with other users:



If you answer "Yes", the name will be saved on the target itself and other users will see it when this target is detected by their RT-LAB. For more information, see [Naming Targets](#).

IP address is the IP address mapped to this target. Change it to map the target to another machine or to change the IP address of the target:



State is the current state of the target (Up or Down).

Operations section

Note: Most of these operations are not available for Windows targets

Operations

- [Shutdown](#) or [Reboot](#) this target.
- Execute a [Python script](#) on this target.
- [Flash](#) an I/O board with a bitstream.
- Execute a custom [command](#).
- [Clean](#) the shared memories.
- [Clean](#) the core dumps.

This section provides some shortcuts relative to the manipulation of the edited target. These operations are also available from the [Target](#) context menu of the [Project Explorer](#).

Shutdown: Shuts down the target. A confirmation dialog is displayed first.

Reboot: Reboots the target. A dialog is displayed first to select the Operating System to be started.

Python script: Opens a dialog to select a Python script. This script may be located in the workspace, on the local file system or on the target. The selected script will then be executed on the target and output will be displayed in the [Target Console](#):

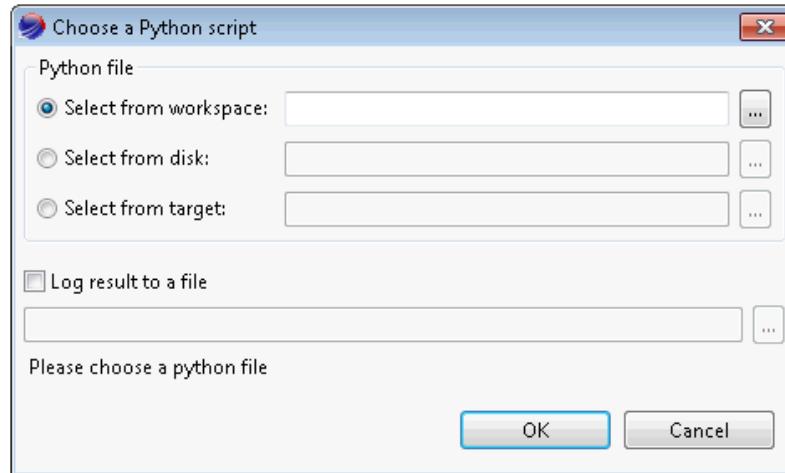


Figure 2: Target Python Script Dialog

Flash bitstream: Opens the [Flash bitstream wizard](#) dialog.

Command...: Opens a simple dialog to enter a command. This command will then be executed on the target and results will be displayed in the [Target Console](#).

Clean shared memory: Cleans the shared memory on the target.

Clean core dumps: Deletes the core dumps on the target.

Diagnostic Page

This page contains two main sections that display detailed information about the target and provide tools to perform a complete diagnostic of the target.

Operating System and Hardware section

Operating System and Hardware	
This section describes detailed information about this target.	
Platform:	QNX 6.x
OS version:	6.3.2
Architecture:	x86pc
Free disk space (MB):	8354
CPU Speed (MHz):	2928
Number of CPU:	2

These properties show the main characteristics of the target. They are also available in the [Properties View](#) when the target is selected in the Project Explorer.

Tools section

Tools

-  [Display](#) I/O boards information.
-  [Display](#) a complete diagnostic.
-  [Open](#) a Telnet terminal.

I/O boards information: Displays information about Opal-RT boards in the [Target Console](#).

Complete diagnostic: Displays advanced hardware and software configuration in the [Target Console](#).

Telnet terminal: Opens the [Terminal View](#) to open a Telnet connection to the target.

Simulation Settings Page

This page contains properties that affect the RT-LAB simulations run on this target.

Note: these properties are not available for Windows targets

Utilities section

Utilities

This section provides some utilities to configure the simulations.

Enable multimodel support

[Remove Embedded Mode](#)

Date:

[Sync. with host](#)

Multimodel support: Allows several models to run on the target at the same time.

Remove embedded mode: Removes any embedded simulation (see [Embedding Simulation](#)) from the target. If a project is currently connected to this simulation, the model will be reset first. If not, a dialog will be displayed to optionally reboot the target.

Date: The current date and time of the target.

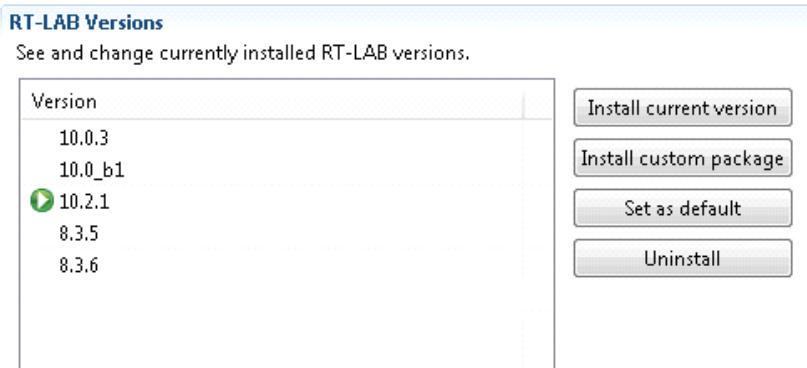
Sync. with host: Set the current date and time of the target to the host computer's ones.

Software Page

This page allows you to install, uninstall and configure softwares on the target.

Note: this page is not available for Windows targets

RT-LAB section

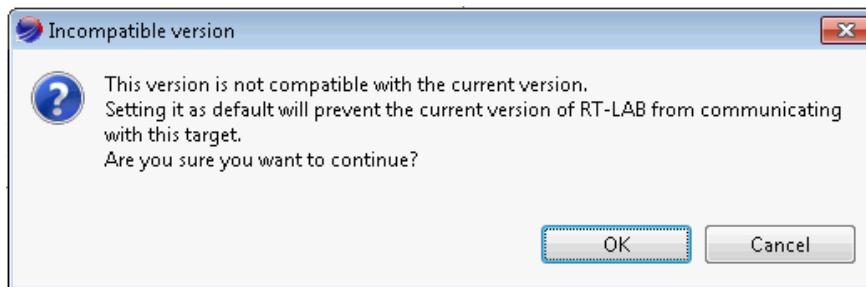


This section shows the list of currently installed versions of RT-LAB. The default one is marked with an icon . This default version is the one that will be used when you compile or load any model on this target so you should make sure that it matches the RT-LAB version running on the host computer.

Install current version: Automatically installs (or re-installs) the current version of RT-LAB on this target. The log will be displayed in the [Target Console](#) and optionally in an editor.

Install custom package: Opens a dialog to select a package from the file system. Packages are .tgz files for QNX and .rpm files for RedHat. The package is then transferred to the target and installed.

Set as default: The selected version will be marked as default. Be careful when you select an old version (e.g. 8.1.x when running 10.x) since you may lose the connection with the target. In this case, a confirmation dialog will be displayed:



Uninstall: Completely removes the selected version from the target and displays the log in the [Target Console](#). If this version was the default one, the most recent version (i.e. the one with the highest version number) will automatically be selected as default.

Views

Views support editors and provide alternative presentations as well as ways to navigate the information in your Workbench. For example, the Project Explorer and other navigation views display projects and other resources that you are working with.

Views also have their own menus. To open the menu for a view, click the icon at the left end of the view's title bar. Some views also have their own toolbars. The actions represented by buttons on view toolbars only affect the items within that view.

A view might appear by itself, or stacked with other views in a tabbed notebook. You can change the layout of a perspective by opening and closing views and by docking them in different positions in the Workbench window.

Related concepts

[Perspectives](#)

[Fast views](#)

[Detached views](#)

[Editors](#)

Fast views

NOTE: The Fast View feature, while still available, has effectively been subsumed by the new minimize behavior (which places minimized stacks into the trim where they subsequently work by showing views in the same manner as the Fast Views). While there's only one Fast View Bar you can have as many minimized stacks as you want and can place them where you want in the trim. See [Maximizing and minimizing elements of the workbench presentation](#) for details.

Fast views are hidden views that can be quickly opened and closed. They work like other views except they do not take up space in your Workbench window.

Fast views are represented by toolbar buttons on the fast view bar, which is the toolbar initially on the bottom left of the Workbench window. When you click the toolbar button for a fast view, that view opens temporarily in the current perspective (overlaying it). As soon as you click outside that view or the view loses focus it is hidden again. The fast view bar can also be docked on the other sides of the Workbench window.

You can create a new fast view by dragging any open view to the fast view bar or by selecting **Fast View** from the menu that opens when you right-click the icon of the view's tab. You can also click on the left-most button of the fast View bar to open a menu which will show a list of views. Selecting a view from this list (or from the dialog that appears if you select '**Other...**' will result in the selected view being added to the fast view bar and activating (i.e. shown as the active part).

Related concepts

[Workbench](#)
[Toolbars](#)

Detached views

Detached views are views that are shown in a separate window with a smaller trim. They work like other views except they are always shown in front of the Workbench window.

You can create a new detached view by dragging any open view outside of the Workbench window or by selecting **Detached** from the menu that opens when you right-click the view's tab.

Related concepts

[Workbench](#)

Project Explorer

This view provides a hierarchical view of the resources in the Workbench.

Here is what the Project Explorer looks like:

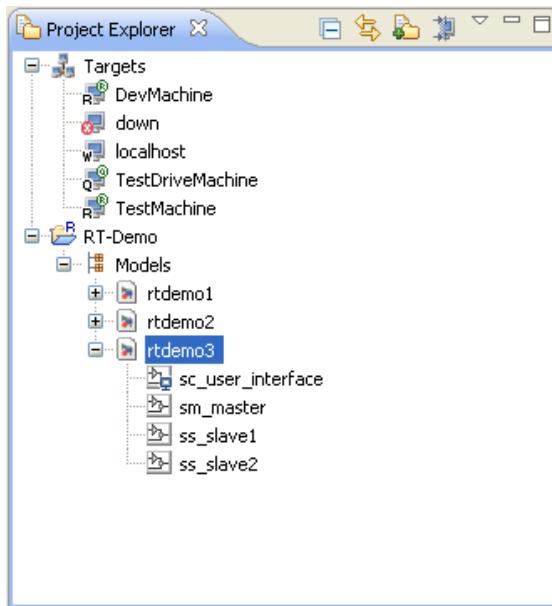


Figure 3:Project Explorer overview

Concepts

The **Project Explorer** is one of the most important parts of the RT-LAB workbench. It allows you to view, create and edit all the resources that you need to complete simulation. This makes the **Project Explorer** a good starting point for any simulation project.

Resources

Basically, there are two kinds of resources: some are specific to RT-LAB (like models and targets) while others are more common resources (like files and folders on the file system).

By default, only RT-LAB resources are shown in the **Project Explorer**. It is possible to display other resources by clicking on the **Filter Resources** button.

Each RT-LAB resource reflects an entity managed by the RT-LAB Controller. To learn more about these resources, see **Organizing resources** and API documentation (sections "RT-LAB Controller Architecture" and "Examples").

Projects

Generally speaking, a project is a set of resources that will interact to achieve a common goal. Several types of projects exist and may appear in the **Project Explorer**. The type of a project is determined at the project creation in the corresponding **New RT-LAB project wizard**.

In order to interact with a simulator, you must create an "RT-LAB" project. This kind of project allows you to group several models or to attach model-related files, such as simulation results or test sequences, to the model itself. This makes it easier to share simulation settings and results.

You may take advantage of other project types if you want to develop specific applications. Project types are generally associated to a particular programming language. For example, create a Python

project if you need advanced Python functionalities or create a C/C++ project if you plan to develop your own S-Function for Simulink.

Each project maps to a directory in the file system. This location is specified when you create a new project. See [New RT-LAB project wizard](#). The content of this directory will be displayed in the [Project Explorer](#) only if you disable the corresponding filter (see [Filter Resources](#) below).

If you are done with a project, you can close it by right-clicking on it and clicking "Close Project". Closed projects require less memory but their contents are not displayed in the [Project Explorer](#). To re-open a project, right-click on it and select "Open Project" or simply double-click on it.

An RT-LAB project can contain any of the resources described below.

Models

Models are the main components of a simulation since they determine the behavior of a simulator. Within the RT-LAB context, the term "model" has to be clarified since it can refer to two distinct elements:

- The **model file**, which is developed with external tools such as Matlab/Simulink or EMTP-RT. It is stored on the file system.
- The **Model object**, or simply Model, which is an interface to the model file in the RT-LAB simulator. It resides in the Controller memory.

Each RT-LAB Model is closely linked to its original model file. For example, a Matlab/Simulink model is saved to a file with the ".mdl" extension. Depending on currently active [Filters](#), this file may appear in the [Project Explorer](#) as shown in [Figure 20](#). Adding this file to an RT-LAB project will create a new Model object which will allow additional interaction such as compilation and execution.

RT-LAB provides a [New RT-LAB model wizard](#) for the creation of a model. Simple templates are available and help you to create a basic skeleton for the real-time subsystems.

This wizard creates both the model file and the corresponding Model object. If you create a model file directly from external tools or if you want to use an existing model file, you must manually create the Model object for the simulation. This can be done using the [Add](#) entry of the [Context menu](#) or the [Drag and Drop](#).

For convenience, all the Model objects that are part of a simulation are displayed as children of a special element named "Models" (See [Figure 3](#)). This element is only a graphical container and is not considered as a resource. Actions that can be handled by this element are described in section named [Model Folder](#) below.

Before launching a simulation, a Model needs to be built as described in [Building models](#). Once built, a Model will contain a list of Subsystems and possibly one Console.

Subsystems

The compilation process creates Subsystems based on the top-level elements of your model file. See [Building models](#).

A Subsystem is a computation unit that can interact with other Subsystems and with the Console.

In order to run a simulation, each subsystem must be assigned to an RT-LAB Target using the [Assignment Page](#).

Console

Like Subsystems, the Console is created during the compilation process. The Console provides basic interaction with the rest of the model: it allows the user to modify the values of control signals and to display the values of acquisition signals.

See [Building models](#) and [Acquiring and Viewing Data](#) for detailed information.

Targets

An RT-LAB Target represents a physical simulator designed to execute Models. They are identified by a name and are associated to an IP address.

Target icons indicate the platform and the state of the Targets. They also allow identification of the different development nodes used for compilation. To manage the Development Nodes, use the [RT-LAB preference page](#) preference page or the context menu of a [Target](#).

ICON	MEANING
	The Target is down or not accessible.
	The Target is up and runs Windows.
	The Target is up and runs QNX.
	The Target is up and runs RedHat.
	The Target is up, runs QNX and is the development node for the QNX platform.
	The Target is up, runs RedHat and is the development node for the RedHat platform.
	The Target is up, runs RedHat and is the development node for several platforms.

Note that there is no Development Node for Windows since the local machine will always be used for compilation.

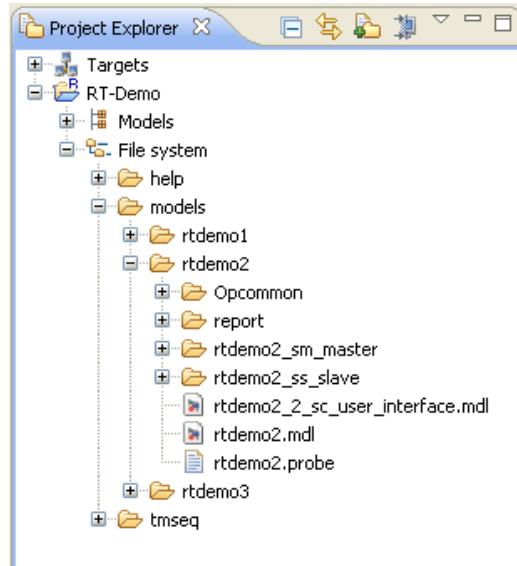
Current platform and Development platform do not need to match because some Targets support multi-boot. For instance, if a multi-boot Target is configured to run both QNX and RedHat and is used as the Development Node for QNX, when it is running RedHat, it remains the Development Node for QNX. In this case, compilation for QNX will not be available. This Target would have the following icon:



As a consequence, multi-boot Targets may be used as Development Nodes for several platforms.

Other resources

The [Project Explorer](#) can be used to manipulate other resources such as files and folders. Except if they are filtered (see [Filters](#) below), all the files and folders present in a project's directory appear as children of the "File system" container:



If you need to use files or folders that are outside of the project's directory, you can use linked resources. To do so, right-click on a project and select **New > File** or **New > Folder**. A dialog is then displayed: click on the "Advanced" button, select "Link to file/folder in the file system" and browse to the file or the folder you want to import.

Here is what this dialog looks like when creating a linked folder:



Figure 4:Linking a folder

The selected folder will then appear in the **Project Explorer** as shown below:

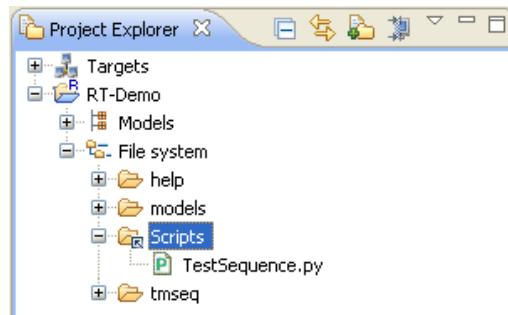


Figure 5: Linked Folder in the Project Explorer

Note that an overlayed arrow on the icon indicates a linked resource.

Since linked resources are only links, deleting such a resource will only remove the link. The original file or folder *will not* be affected. However, children of a Linked Folder are *not* linked so deleting them *will* destroy the original files or folders.

For further documentation, see the following paragraph and **Workbench User Guide > Concepts > Workbench > Linked resources**.

Organizing resources

Resources on the file system

As said above, each project maps to a directory on the file system. By default, these directories are located in the workspace folder (<RTLAB_ROOT>/workspace/). A project's directory has the same name as the project itself, so renaming a project in the RT-LAB Workbench will rename its directory automatically. Similarly, deleting a project will entirely delete its directory.

When you create a resource in a project, for example when creating a new Model with the **New RT-LAB model wizard**, this resource is stored in its mapped directory.

The default resource hierarchy of a project respects the following rules:

- There is one folder per Model, containing all model-related files
- All Model folders belong to a folder named "models"
- This "models" folder is a direct child of the project

Here is an example of this hierarchy:

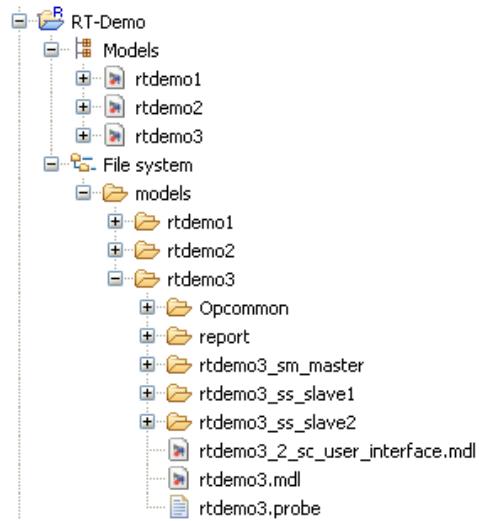


Figure 6:Standard resource hierarchy

Working with linked resources

Although respecting the previously described resource hierarchy is highly recommended for an easier resource management, sometimes it may be useful to manipulate resources that are located outside of a project's directory. This is the case when several users want to share some files through a network or when several projects use the same files.

To add external files and folders to a project, you have to create linked resources, as described in the [Other resources](#) section.

Project location

By default, a project is created in the current workspace directory but you can specify another location using the [New RT-LAB project wizard](#). This is useful to add an existing project that has been created by another application. This is automatically done when you add [Active Projects](#) to the workspace.

Be very careful when creating such projects, since all existing files and folders that already are in the project's location will belong to this project. Deleting this project will delete *all* its files, even if you set the project's location to "C:\\" or to other sensible folder!

Model location

Models are always located in their project's directory when they are created with the [New RT-LAB model wizard](#).

However, you can add an existing Model with the [Add Model wizard](#). If the model file is located outside of the project's directory, a linked resource is automatically created.

Best practices

- As said before, except in a few cases, project and model files should be created accordingly to the default hierarchy. To add existing projects or Models to the RT-LAB workbench, prefer using the [Existing RT-LAB model import wizard](#) that will make a copy of the files to the RT-LAB workspace.
- To avoid conflicts between files, do not put several model files into the same folder.
- It is not recommended to use more than ten Models within the same project. As Models objects provide a lot of functionalities, they are quite resource consuming. Using many Models at the same time can lead to a significant slowdown of the system or to some unexpected behaviors.

If a Model is temporarily not used, remove it from its project (without deleting its model file), then re-add it when necessary. If you manipulate many Models, for example if you have many versions of a Model, consider using one project per Model instead of adding all Models to the same project.

- For the same reasons, unused projects should be closed to improve global performances.

Model and project states

Models

A Model may be in different states depending on the actions performed by the user on it. The typical state sequence is: Not Loadable, *Compiling*, Loadable, *Loading*, Paused, Running, *Resetting*, Loadable.

Compiling, *Loading* and *Resetting* are transitional states. During these states, there are very few possible interactions with the model.

When a Model is in one of the following states: Compiling, Loading, Paused, Running or Resetting, its state is displayed as shown below:



Figure 7: Model state: running

Projects

When an RT-LAB project is closed or collapsed, its state is displayed in three cases:

- If at least one of its Models is currently Loading, Paused, Running or Resetting, the project is in the global state "Running":



Figure 8: Project state: running

- If no Models are running and if another application, such as a Python script, is connected to this project, its state is set to "Active":



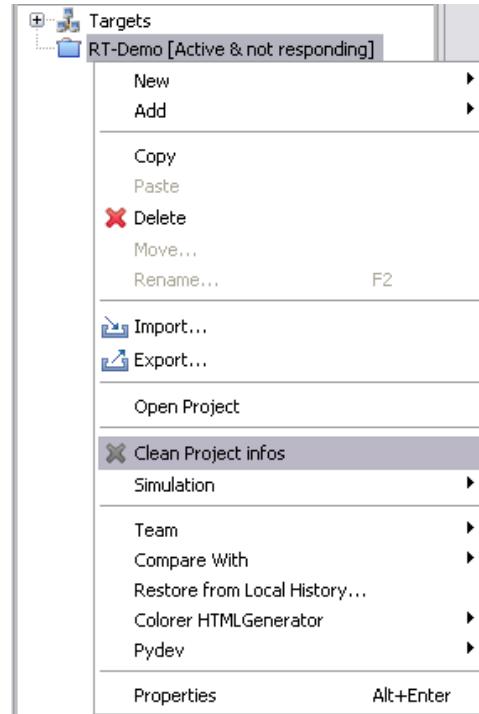
Figure 9: Project state: Active

- If at least one of its Model was Loading, Paused, Running or Resetting, but does not respond anymore, its state is set to "Active & not responding":



Figure 10: Project state: Active but not responding

When an RT-LAB project is "Active & not responding", it cannot be opened. If for some reason, one of its models is definitely lost (for example, its target has crashed), you can right-click on this project and select "Clean Project infos" as shown below:



This will reset the state of unreachable models and will allow the project to be opened again.

Label Decorations

Decorations display additional information on the elements of the **Project Explorer** by adding an overlay icon or a short suffix. They are used for example to display **Model and project states** or to show the platform of a Target.

All these decorations are optional and can be disabled in the [Label Decorations preference page](#).

The **Project Explorer**'s menu provides quick access to the following decorations:

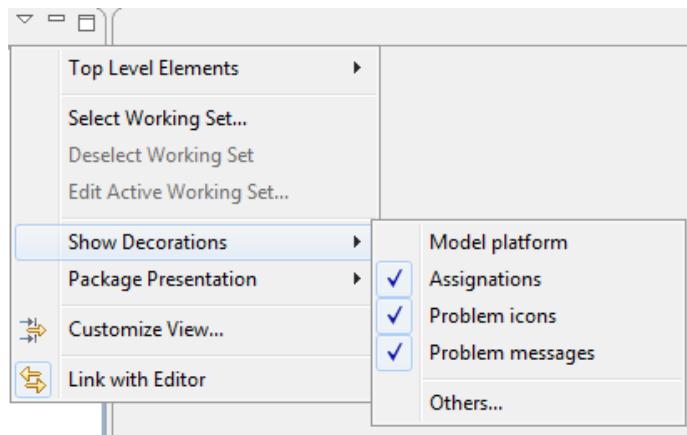
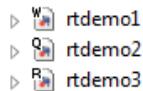


Figure 11: View menu: Show Decorations

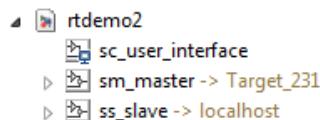
Model platform

Adds an overlay icon to Models to show their current platform:



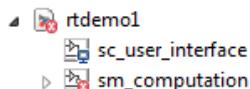
Subsystem Assignment

Adds the name of the assigned target next to models (if applicable) or subsystems:



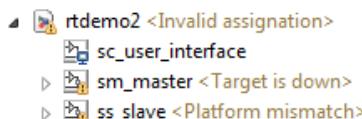
Problem icons

Adds an overlay icon to RT-LAB items to show errors and warnings:



Problem messages

Adds a message next to RT-LAB items to describe errors and warnings:



Others... (needs the Advanced Capability)

Opens the [Label Decorations preference page](#).

Active Projects

The [Project Explorer](#) is not the only way to create RT-LAB projects. For example, a Python script can use `RtLabApi.NewProject()` to create its own RT-LAB project. Such projects are displayed in the [Project Explorer](#) as shown below:

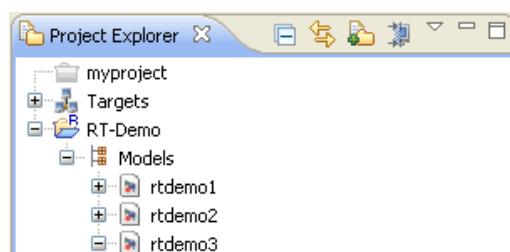


Figure 12: Active Project

"myproject" is not part of the workspace because it has been created by an external application. Note that it cannot be expanded and there is no possible interaction with it.

If you want to see the content of this project and to interact with it, you need to import it to your Workspace. The easiest way to realize this is to right-click on the Active Project and select "Add to workspace". The Active Project will then be replaced by a standard RT-LAB project:

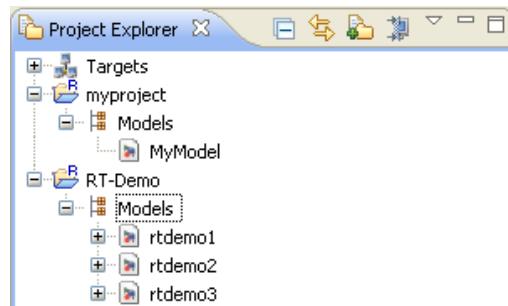


Figure 13:Imported Active Project

Working sets

Working sets are useful to organize projects into groups. This is very helpful to identify related projects or when the Workspace contains a lot of projects.

Working sets are managed through the View menu:

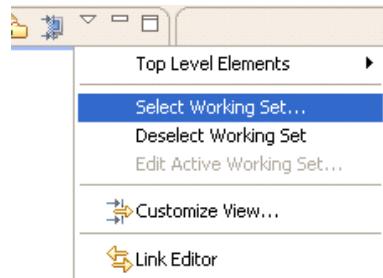


Figure 14:Working Set Menu

Clicking on "Select Working Set..." displays the following dialog:

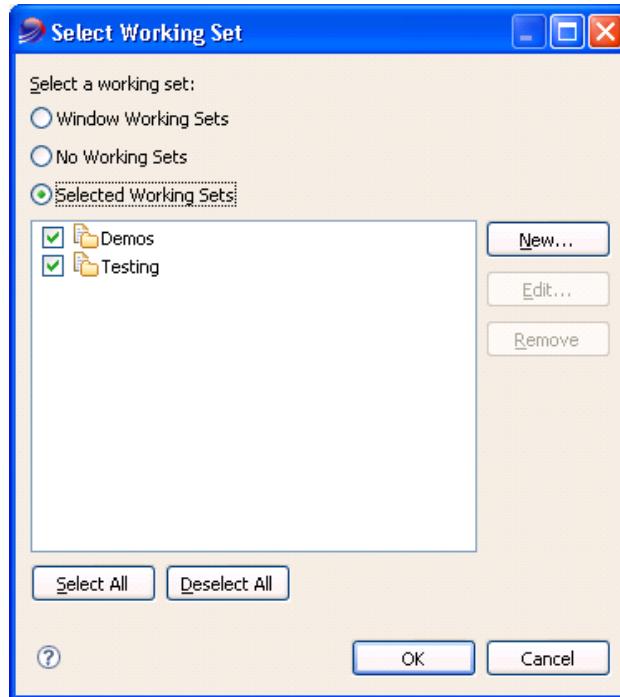


Figure 15:Working Set Dialog

In this dialog, you can create, edit and select Working sets. Editing a Working set allows you to choose which projects are part of this Working set. Note that a project may belong to several Working sets.

Working sets may affect the presentation of the **Project Explorer** in two different ways, depending on the "Top Level Elements" menu:

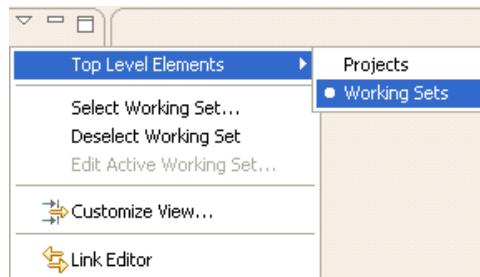


Figure 16:Top Level Element menu

If "Working Sets" is selected, the selected Working sets appear in the **Project Explorer** as containers:

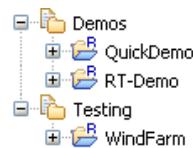


Figure 17:Working Set display

If "Projects" is selected, only projects that are part of the currently selected Working sets will appear in the **Project Explorer**. Other projects will be hidden.

Clicking "Deselect Working Set" makes all existing projects appear in the **Project Explorer** again.

Filters

The **Project Explorer** uses Filters to determine what types of resources must be displayed. In other words, applying a Filter makes the corresponding resources disappear from the **Project Explorer**.

To access Filters, choose "Customize View..." from the View Menu:

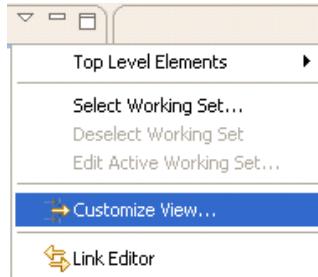


Figure 18:Customize View menu

The following dialog is displayed:

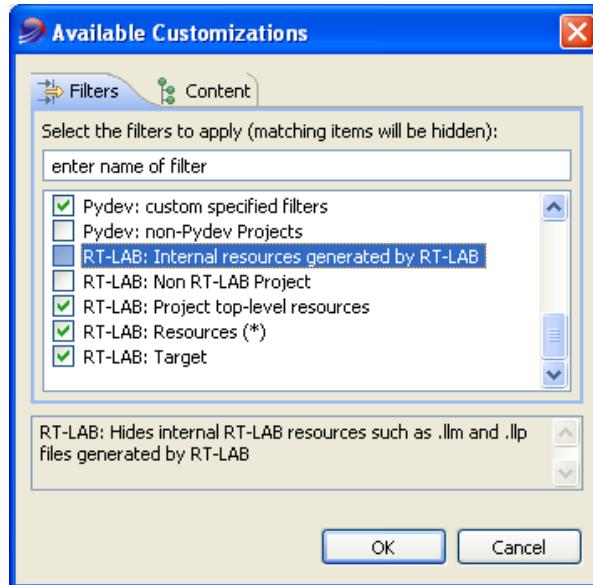


Figure 19:Customize View (Filters) dialog

Here is a brief description of the RT-LAB filters:

FILTER	DESCRIPTION
Internal resources generated by RT-LAB	Hides internal files such as .llm, .llp... Note: This has no effect if the <i>Resources</i> filter is active.
Non RT-LAB Projects	Shows RT-LAB projects only.
Projects top-level resources	Hides top-level resources and displays them in the "File system" container.
Resources	Hides non RT-LAB resources from RT-LAB projects. Note: A shortcut is provided on the toolbar for this filter (see Filter Resources).

Target	Shows only Targets that match user settings. Note: Settings are editable through the ToolBar (see Filter Targets).
---------------	--

ToolBar

The Toolbar of the [Project Explorer](#) contains the following elements:

ICON	NAME
	Collapse All
	Link Editor
	Filter Resources
	Filter Targets
	View Menu

Collapse All

This command collapses the tree expansion state of all resources in the view.

Link Editor

This command toggles whether the [Project Explorer](#) view selection is linked to the active editor. When this option is selected, changing the active editor will automatically update the [Project Explorer](#) selection to the resource being edited.

Filter Resources

This command toggles whether the “Resources” filter is applied or not.

When this filter is enabled, the files and the folders that are part of an RT-LAB project are hidden from the [Project Explorer](#), as shown in [Figure 3](#).

When this filter is disabled, files and folders are visible:

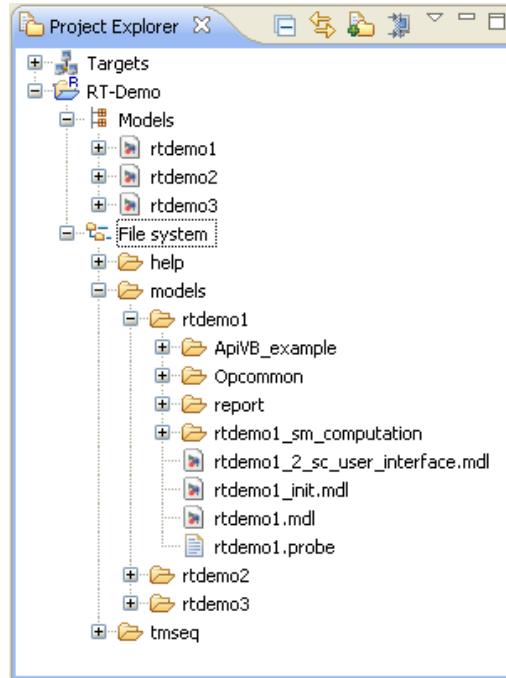


Figure 20:Project Explorer with Resources

Note that some resources may remain hidden if other filters are active, such as the "Internal files" filter.

Filter Targets

This command toggles whether the Target filter is applied or not and allows the user to configure this filter.

Clicking on this button displays the following dialog:

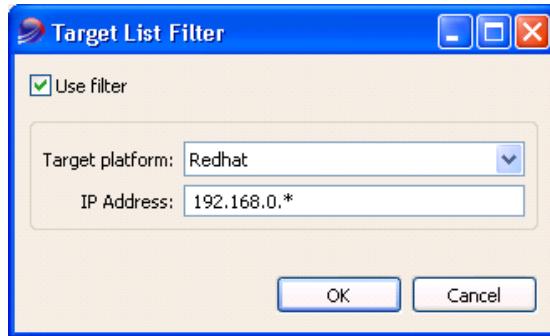


Figure 21:Target Filter dialog

In this example, only the Targets that run RedHat and whose IP address starts with 192.168.0. will be displayed in the [Project Explorer](#).

View Menu

This command displays a menu for managing [Working sets](#) and [Filters](#).

Context menu

A Context Menu is displayed whenever a right-click is performed on the [Project Explorer](#). The content of this menu depends on what element is selected and what [Capabilities preference page](#) are enabled.

Common

Here are the menu entries that are common for files, folders, projects and some other resources:

New

Enables you to create new resources by opening the corresponding wizard. Note that before you can create a new file, you must create a project in which to store the file.

Copy (Ctrl+C)

This command places a copy of the selection on the clipboard.

Paste (Ctrl+V)

This command places the text or object on the clipboard at the current cursor location in the currently active view or editor.

Delete (DEL)

This command removes the current selection.

Rename (F2)

Enables you to change the name of the currently selected resource.

Import

Launches the Import wizard, which enables you to add resources to the Workbench.

Export

Launches the Export wizard, which enables you to export resources from the Workbench.

Refresh (F5)

Refreshes the resource with the contents in the file system.

Properties

Opens the Properties dialog for the currently selected resource.

File

Here are the menu entries for file resources:

Open

Opens the file in its associated editor.

Open With

Opens the file with the specified editor.

Close

Closes the active editor. You are prompted to save changes before the file closes.

Project

Here are the menu entries for project resources:

Close Project

Closes this project. See [Organizing resources](#).

Close Unrelated Projects

Closes all other projects that are not related to this project.

RT-LAB Project

Here are the menu entries for RT-LAB project resources:

Add

Opens the [Add Model wizard](#) to add an existing model to this RT-LAB project.

Clean Project infos

Available when the project is in "Active & not responding" state (see [Model and project states](#)).

Clears execution information about models that are unreachable (for example, when a target has been shut down). This will allow the project to be opened again.

Simulation

Provides access to simulation commands such as Load, Execute, Reset... See [Simulation menu](#). These commands will be applied to the entire Project, it means to all the Model objects that are part of this RT-LAB project.

Properties

Opens the [Properties View](#) to display informations about this RT-LAB project.

Active Project

Here are the menu entries for Active Project resources:

Add to workspace

Imports this Active Project to the workspace. This will enable interaction with this project. See [Active Projects](#)

Clean Project infos

Available when the project is in "Active & not responding" state (see [Model and project states](#)).

Clears execution information about models that are unreachable (for example, when a target has been shut down). This will allow the project to be opened again.

Model Folder

Here are the menu entries for Model Folder resources:

New

Opens the [New RT-LAB model wizard](#) to create a new Model.

Add

Opens the [Add Model wizard](#) to add an existing model.

Simulation

Provides access to simulation commands such as Load, Execute, Reset... See [Simulation menu](#). These commands will be applied to all children Models.

Model

Here are the menu entries for Model objects:

New

Opens the [New RT-LAB model wizard](#) to create a new Model.

Add

Opens the [Add Model wizard](#) to add an existing model to the parent project.

Open

Open the editor associated with this Model

Edit

Edit this Model with Matlab/Simulink.

Edit With...

Edit this Model with the specified version of Matlab/Simulink.

Note that the selected version will become the default Matlab version for the entire workbench. See [Matlab preference page](#) preference page to edit this setting.

Simulation

Provides access to simulation commands such as Load, Execute, Reset... See [Simulation menu](#).

Properties

Opens the [Properties View](#) to display informations about this Model.

Model file

In addition to file-related menu entries, here are the menu entries for Model file:

Add to Project (only in RT-LAB projects)

Add this model file to its parent RT-LAB project. That will create an associated Model object (See [Model](#)).

If this model file has already been added to this RT-LAB project, the menu is grayed and disabled.

Note that this action will be directly executed, without opening the [Add Model wizard](#).

This action may also be done using [Drag and Drop](#).

Subsystem

Here are the menu entries for Real-Time Subsystems:

Simulation

Provides access to simulation commands such as Load, Execute, Reset... See [Simulation menu](#). These commands will apply to the parent Model.

Properties

Opens the [Properties View](#) to display informations about this Subsystem.

Console

Here are the menu entries for Console Subsystems:

Edit (Simulink models only)

Opens this Console for edition in Matlab/Simulink.

Target Folder

Here are the menu entries for Target Folder:

New

- **New Target:** Opens the [New Target wizard](#).

Discover Targets

Performs a target detection over the network and opens the [Detected Targets Wizard](#).

Target

Here are the menu entries for Target (Note that some entries may not be available depending on the current platform of the selected Target):

New

- **New Target:** Opens the [New Target wizard](#).

Install

- **RT-LAB:** Automatically install (or re-install) the current version of RT-LAB on this Target.
- **Patch...:** Opens a dialog to select a Patch file to be sent and installed on this Target. Patch files are either .rpm files for RedHat Targets, either .tgz files for QNX Targets.

Execute

- **Shutdown:** Shutdown this Target. A confirmation dialog is displayed first.
- **Reboot:** Reboot this Target. A dialog is displayed first to select the Operating System to be started.
- **Remove Embedded Mode:** If an [Embedding Simulation](#) is running, this will prevent it from starting at the next Target power up.
- **Python script:** Opens a dialog to select a Python script. This script may be located in the workspace, on the local file system or on the Target. The selected script will then be executed on the Target and output will be displayed in the [Target Console](#):

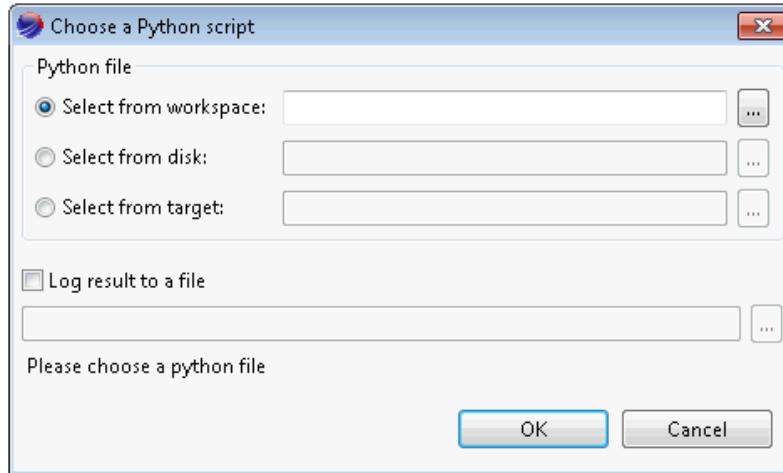


Figure 22: Target Python Script Dialog

- **Flash bitstream:** Opens the [Flash bitstream wizard](#) dialog.
- **Command...:** Opens a simple dialog to enter a command. This command will then be executed on this Target and results will be displayed in the [Target Console](#).
- **Clean shared memory:** Cleans the shared memory on this Target.
- **Clean core dumps:** Cleans the core dumps on this Target.

Tools

- **Get I/O Infos:** Displays informations about this Target's I/O cards in the [Target Console](#).
- **Diagnostic:** Displays a detailed diagnostic in the [Target Console](#).
- **Telnet:** Opens the [Terminal View](#) to open a Telnet connection to this Target.

Set As Development Node

The selected Target will become the development node for its current platform. By default, the compilation of any Model running on this platform will be done on this Target.

This setting may also be changed in [RT-LAB preference page](#).

Properties

Opens the [Properties View](#) to display informations about this Target.

Embedded Simulation (any item)

Connect

Opens the [Connect to Embedded Simulation Wizard](#) to reconnect a project to this [Embedding Simulation](#).

Drag and Drop

Using the Drag and Drop functionality is a fast and easy way to accomplish some usual actions. The following table provides a description of possible actions depending on what resource is dragged on what destination.

SOURCE	DESTINATION	ACTION
File	Folder	Move this file to the destination folder. Hold <Ctrl> key to make a copy instead of moving the file.
File or folder	Project	Move this file or folder to the destination project. Hold <Ctrl> key to make a copy instead of moving the file or folder.
Model or model file	RT-LAB project or Models folder	Add the model file to the destination project and create a new Model resource in this project. Hold <Ctrl> key to make a copy of the model and add this copy to the project.
Model or model file	Target	Assign all the subsystems of this model to this Target.
Model or model file	Editors area	Open an Editor to edit the properties of this model.
Model or model file	Matlab View	Open this model in Simulink.
Subsystem	Target	Assign this Subsystem to this Target.
Subsystem	Editors area	Open an Editor to edit the properties of the parent model.
Subsystem	Matlab View	Open the parent model in Simulink.
Console	Matlab View	Open this Console in Simulink.
Target	Subsystem	Assign this Subsystem to this Target.
Target	Model	Assign all the subsystems of this model to this Target.
Target	Target folder	Hold <Ctrl> key to make a copy of this Target.
Python script (*.py)	Target	Run this script on the destination Target. Results will be displayed in the Target Console .
Bitstream (*.bin)	Target	Opens the Flash bitstream wizard dialog to flash a board with this bitstream.

Related references

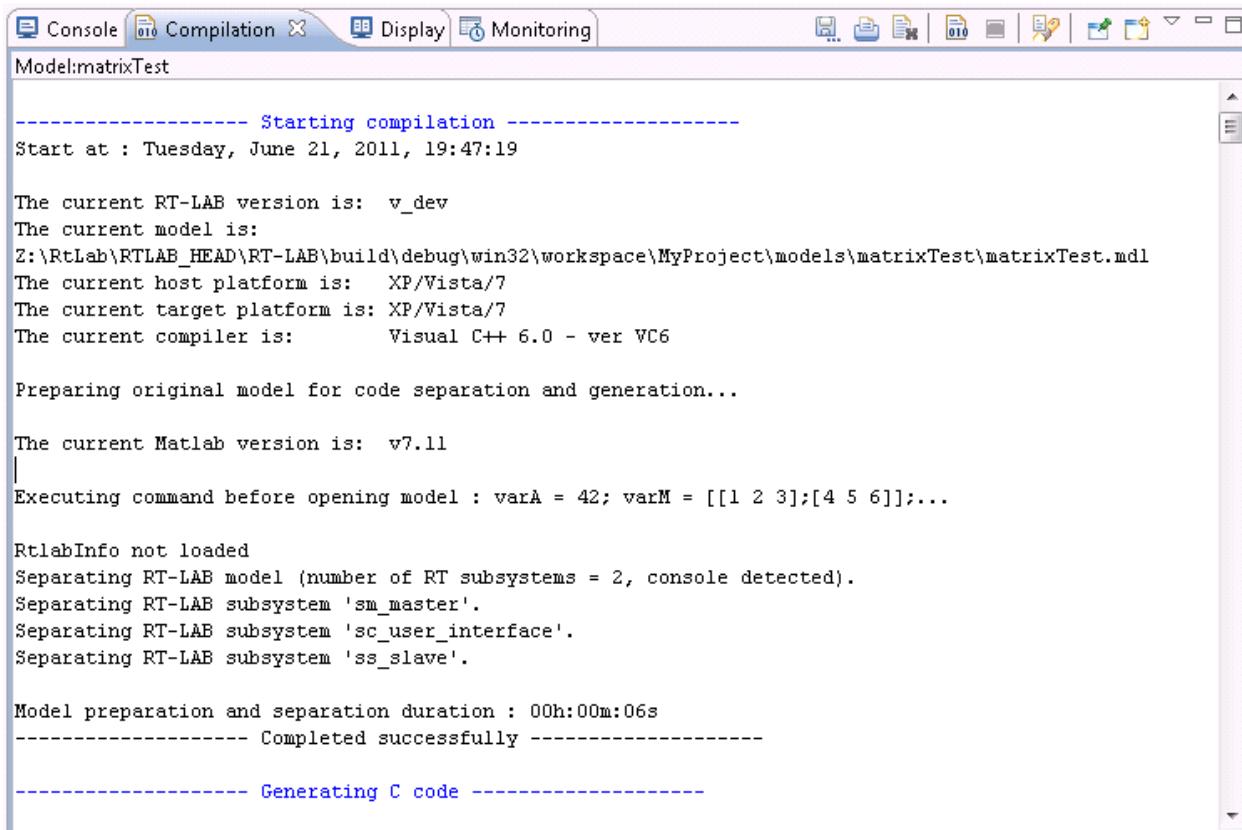
- [Wizards](#)
- [Executing models](#)
- [Toolbars](#)
- [Workbench Menus](#)

Compilation View

The **Compilation View** displays the compilation output generated while building a model with RT-LAB. The result gives important information about the model's compilation. Note that the **Compilation View** highlights errors and warnings with red and orange colors respectively. Indeed, a model that is not properly built is not executable on the target nodes.

To start the compilation of a model, it is possible to click on the Build button in the Overview tab or by right-clicking on the model and selecting Build from the Simulation menu. For more information, please see [Building models](#).

The **Compilation View** appears in a tabbed notebook:



The screenshot shows the RT-LAB interface with the 'Compilation' tab selected in the top navigation bar. The main area displays the compilation log for a model named 'matrixTest'. The log includes the following text:

```

----- Starting compilation -----
Start at : Tuesday, June 21, 2011, 19:47:19

The current RT-LAB version is: v_dev
The current model is:
Z:\RtLab\RTLAB_HEAD\RT-LAB\build\debug\win32\workspace\MyProject\models\matrixTest\matrixTest.mdl
The current host platform is: XP/Vista/7
The current target platform is: XP/Vista/7
The current compiler is: Visual C++ 6.0 - ver VC6

Preparing original model for code separation and generation...

The current Matlab version is: v7.11
|
Executing command before opening model : varA = 42; varM = [[1 2 3];[4 5 6]];...

RtlabInfo not loaded
Separating RT-LAB model (number of RT subsystems = 2, console detected).
Separating RT-LAB subsystem 'sm_master'.
Separating RT-LAB subsystem 'sc_user_interface'.
Separating RT-LAB subsystem 'ss_slave'.

Model preparation and separation duration : 00h:00m:06s
----- Completed successfully -----

----- Generating C code -----

```

Figure 23:Compilation View

The **Compilation View** has three important elements: selected model name, compilation output and a toolbar.

The name of the selected model is displayed at the top left of the **Compilation View**. This information is available only if a model is selected in the **Project Explorer** view. The name of the model in the above figure is: rtdemo2

By default, the **Compilation View** is included in the Edition perspective. To add it to the current perspective, click Window > Show view > Other... > RT-LAB > Compilation

Compilation output

The **Compilation View** provides the following information while compiling.

Starting compilation information:

- Start time: Timestamp of when the compilation has started
- RT-LAB version: Current RT-LAB GUI version
- Model: Current path of the model on the host computer
- Host platform: Name of the current host platform
- Target platform: Name of the current target platform
- Current Matlab version: Current version of Matlab used to build the model
- Model Matlab version: Initial version of Matlab used to create the current model
- Model preparation and separation
- Successful state: Shows if the model preparation and separation has been completed successfully

C code generation information:

- Displays the output of the Real-Time Workshop C code generation for each subsystem and prints the specific commands that are invoked on the target OS.

Database information:

- Displays the database output for parameters and signals.

Copying the generated C code:

- Outputs the destination directories and shows which files are transferred from host to target via FTP protocol.

Building master and slave subsystems:

- Displays, for each real-time subsystems, the results of the compilation and the linking of the C code into an executable file.

Transferring the built model:

- Outputs which files are retrieved from target to host via FTP protocol.

Completion information:

- End time: Timestamp of when the compilation has ended
- Compilation duration: Difference between end time and start time

The compilation output also displays the warnings and errors that occurred during the build process. Errors report problems serious enough to stop the compilation process while warnings report other unusual conditions that may lead to further problems. Warnings will not stop the compilation process though.

Toolbar

The toolbar in the **Compilation View** contains the following buttons:

ICON	DESCRIPTION
	Save As Saves the current output of the compilation to a specified text file.
	Print Prints the current compilation output.
	Clear Log Clears the current compilation output.
	Build Starts a new build process. Note that the previous compilation output will be cleared before printing the new compilation output.
	Abort build Stops the current building process.
	Search text Searches specified text in the compilation output.
	Pin view to selection Pins the view to selection means that the Compilation View will fix the output to the current selected model. So, the output of the pinned Compilation View will not be updated if you select another model from the Project Explorer view and build it.
	New View Creates a new Compilation View that will be stacked in the tabbed notebook at the rightmost position This command can be useful when you want to pin views on different models.
	View Menu

View Menu

The **Compilation View** provides menu items that allow you to activate the Auto scrolling and Word wrap options for the compilation output.

Auto scrolling

This menu item enables/disables the automatic scrolling of the window when text is added. When enabled and text added, the view automatically goes to the bottom of the window where the latest text is.

Word wrap

This menu item enables/disables the wrapping of the text to the window size. It allows to displaying a text on a new line when the line is full, such that each line fits in the viewable window.

Related concepts

- [Views](#)
- [Perspectives](#)
- [Toolbars](#)

Related tasks

[Building models](#)
[Opening views](#)
[Moving and docking views](#)

Related reference

[Project Explorer](#)

Display View

The **Display View** provides information about the operations that are performed on the model during its life cycle. From the output, you can identify in which state the model is: not loaded, running, paused or reset.

The **Display View** appears in a tabbed notebook:

```

matrixTest / sm_master
Model 'matrixtest_1_sm_master' compiled in RELEASE mode.
2 CPUs active on this Computer
libOpal.a (Simulink - R2010B): v_dev (DEBUG MODE, May 2 2011, 15:24:10)
core count
Subsystem sm_master allocates 1 cores.
Subsystem ss_slave allocates 1 cores.
model matrixtest_1_sm_master assigned to logical cpu 1
Monitoring is enabled
RECV: connection to host established
SEND: connection to host established
A Unit delay is applied on status exchange.
Display of standard output will be disabled
Monitoring: start time = 0.000 ms, using CPU speed = 2933 MHz
SubSystem step size = 0.001000 sec. Status updated at every 1 local step.
Synchronized with software timer.
Real-time SingleTasking mode.
Snapshot taken (opmatrixtest_sm_master_0.snap).
[0]: PAUSE mode, IO set to pause value.
    Total of 0 Overrun detected.
    Tue Jun 21 18:34:24 2011

[1]: RUN mode, IO set to run value.
    Synchronized step size = 1000 us.
    Tue Jun 21 18:34:26 2011

```

Figure 24:Display View

The **Display View** has three important elements: selected (model / subsystem) names, display output and toolbar.

The name of the selected (model / subsystem) is showed at top left of the **Display View**. This information is available only if a subsystem is selected in the **Project Explorer** view. To see all subsystems, you need to expand the model tree and select the subsystem items. Subsystem name starts with sm for master or ss for slave. The name of the (model / subsystem) in the above figure is: rtdemo2 / sm_controller. By default, when selecting a model, the view will show information related to the master subsystem.

The subsystem with the name that starts with sc is the console subsystem. The console subsystem is running in asynchronous mode and usually contains controls and displays to drive/monitor the simulation. No information is provided for this subsystem by the **Display View**. For more information, please see **Using the Console**.

The **Display View** is useful only after the model has been built. For more details on compilation information, please see **Compilation View**.

By default, the **Display View** is included in the Edition perspective. To add it to the current perspective, click Window > Show view > Other... > RT-LAB > Display

Display output

Each model presents one display output for each real-time subsystems.

This section resumes the information outputs, available for each real-time subsystems, when executing the basic operations on a model.

Note that the display output highlights errors and warnings with red and orange colors respectively.

Loading a model

When loading a model, real-time executables are loaded on the target nodes. For each subsystem, the [Display View](#) is updated during this operation.

For the master and slave subsystems includes:

- Files transferred to the target nodes
- Name of the executable
- Number of active CPUs on the computer
- Library used by RT-LAB
- RT-LAB version and build tag
- Monitoring state
- Connection state with the host
- CPU speed of the target nodes
- Subsystem step size
- Real-Time simulation mode
- Snapshot status

Executing a model

When executing a model, the simulation starts on the target nodes. For each subsystem, the [Display View](#) is updated during this operation.

Display information for the master and slave subsystem:

- Synchronized step
- Timestamp of when the model has been started

Pausing a model

When pausing a model, the simulation of the model is paused on the target nodes and waits until a subsequent execution is performed. For each subsystem, the [Display View](#) is updated during this operation.

Display information for the master and slave subsystem:

- Number of overruns detected during simulation
- Timestamp of when the model has been paused

Resetting a model

When resetting a model, the simulation of the model is stopped. For each subsystem, the [Display View](#) is updated during this operation.

Display information for the master and slave subsystem:

- Number of overruns detected during simulation
- Timestamp of when the model has been reset
- File transfer of the generated result files to the command station

Adjusting a model during simulation

When adjusting a model during simulation, the display view is updated and indicates the changes that occurred on the model. For example, when modifying some parameter values, the display view indicates that some parameters have been updated.

The display provides many other useful information when actions are performed on the simulation or when changes occur on the model.

Toolbar

The toolbar in the [Display View](#) contains the following buttons:

ICON	DESCRIPTION
	Save As Saves the current output of the display to a specified text file.
	Print Prints the current display output.
	Clear Log Clears the current display output.
	Search text Searches specified text in the display output. See figure: Find dialog.
	Pin view to selection Pins the view to selection means that the Display View will lock the output to the current selected model / subsystem of the project explorer. So, the view will continue to display the output of the selected model / subsystem even if the user selects another model / subsystem from the Project Explorer view.
	New Display Creates a new Display View that will be stacked in the tabbed notebook at the rightmost position. This command can be useful when you want to pin views on different subsystems or if you want to compare displays.
	Horizontal Provides organization of the Display View by placing outputs in horizontal strips, one above the other. You need to select more than one subsystem simultaneously to see the outputs tiling. To select more than one subsystem, go to the Project Explorer view, expand the model and press Ctrl + click on the sm and ss subsystems.

Vertical
 Provides organization of the [Display View](#) by placing outputs in vertical strips, side by side. You need to select more than one subsystem simultaneously to see the outputs tiling. To select more than one subsystem, go to the [Project Explorer](#) view, expand the model and press Ctrl + click on the sm and ss subsystems.

View Menu

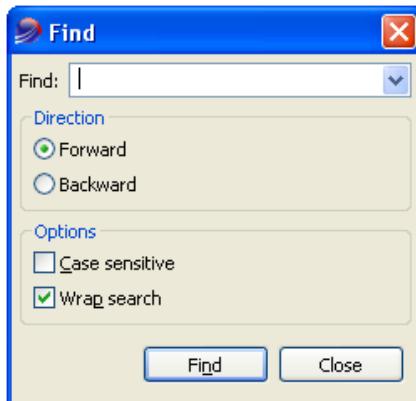


Figure 25:Find dialog

View Menu

The [Display View](#) provides menu items that allow you to activate the Auto scrolling and Word wrap options for the display output.

Auto scrolling

This menu item enables/disables the automatic scrolling of the window when text is added. When enabled and text added, the view automatically goes to the bottom of the window where the latest text is.

Word wrap

This menu item enables/disables the wrapping of the text to the window size. It allows to displaying a text on a new line when the line is full, such that each line fits in the viewable window.

Related concepts

[Views](#)
[Perspectives](#)
[Toolbars](#)

Related tasks

[Loading the model](#)
[Running the model](#)
[Resetting the model](#)
[Using the Console](#)
[Opening views](#)

Moving and docking views

Related reference

[Project Explorer](#) view

Variable Viewer

The **Variable Viewer** displays information about any selected variable in your model. It also lets you edit model variables. A variable can be a signal input, parameter or any MATLAB variable in your model. The **Variable Viewer** appears in a tabbed notebook.

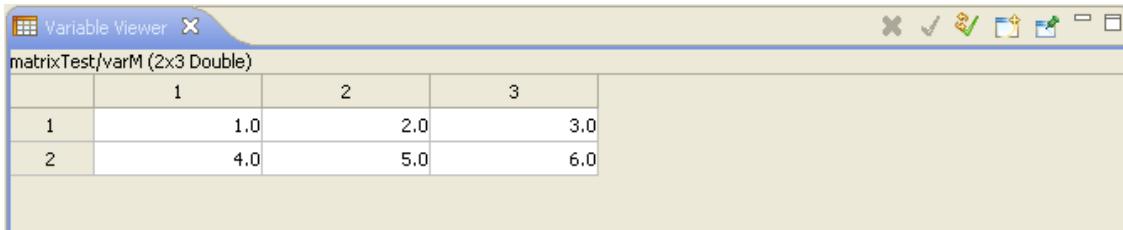


Figure 26:Variable Viewer

By default, the **Variable Viewer** is included in the Edition perspective. To add it to the current perspective, click Window > Show view > Other... > RT-LAB > Variable Viewer.

The **Variable Viewer** view can also be obtained for a variable, by expanding the model in the Project Explorer, selecting a variable and double-clicking on it. A variable can also be displayed by expanding the project explorer tree, right-clicking on the variable and selecting the tab "Show-In".

The **Variable Viewer** supports multiple views. A current view can be pinned to the tab in order to allows the display and /or the edition of another variable in others views.

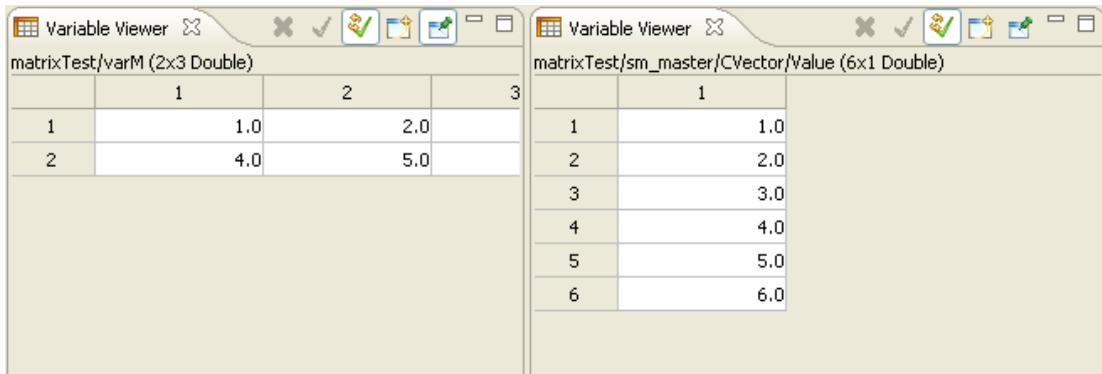


Figure 27:Variable Viewer: Multiple views with pinned one

Editing values

Variable Viewer is useful only if a model is loaded and information is available if a variable is selected. The name of the selected variable is showed at top left of the **Variable Viewer**.

Variable Viewer offers three possibilities: Apply Changes, Apply Changes immediately and Discard Changes. These buttons allow to edit or discard changes from a variable.

In auto-apply mode (the default mode), triggered by **Apply changes immediately** toggle button any change in the value of variables is automatically taken into account , when enter keyboard is pressed.

In non auto-apply mode, (the **Apply changes immediately** button is unchecked), a value can be edited without being sent to the model. Once edited the value can be applied or discarded and changes cancelled. An asterisk appears after a value when it has been edited but the changes have not yet been applied .

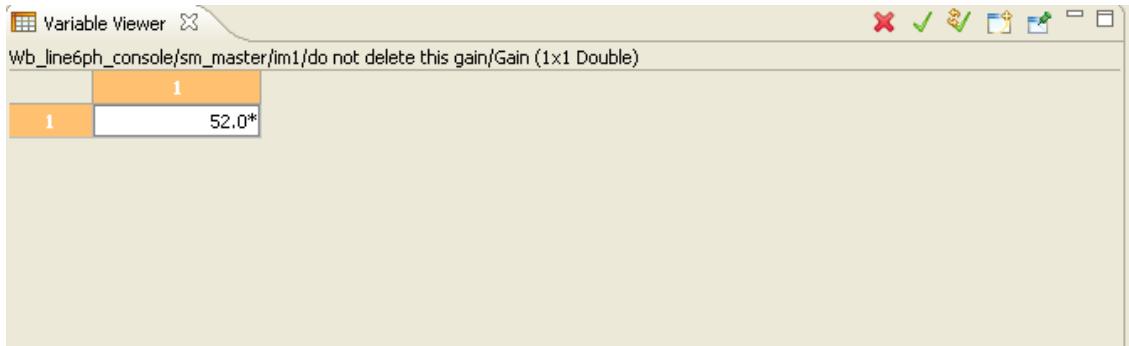


Figure 28: Viewer explorer value being edited

The **Apply Changes** button allows you to confirm the changes. Once the change is accepted , the asterisk disappears.

The **Discard Changes** allows you to return to the original value. Note that this change must be made before any further changes are accepted

To display multiple variable simultaneously, and for more details to the multiple view display, please see : Variable Table

Display output highlights data format error with red color.

Toolbar

The toolbar in the **Variable Viewer** contains the following buttons:

ICON	DESCRIPTION
Pin view	Pins the view to selection means that the Variable Viewer will lock the output to the current selected variable, so the view will continue to display the output of the selected variable.
New View	Creates a new Variable Viewer that will be stacked in the tabbed notebook at the rightmost position. This command can be useful when you want to pin views on different variables or if you want to compare displays.
Auto-Apply (Apply changes immediately)	Allows the changes for any selected variable to be applied immediately just after typing the edited value and clicking enter, or after the mouse loose focus. Changes are propagated to certain views like Properties or Variable Table.
Apply Changes	Provides edition for provisory changes to the selected variable of the Variable Viewer . When in non-auto apply modes, changes are not taken into account. Values are displayed with a star above. The changes made can be accepted or rejected.
Discard Changes	When in non-auto apply mode changes are not taken into account immediately. Edited changes can be discarded with the discard button

Related concepts

- [Views](#)
- [Perspectives](#)
- [Toolbars](#)

Related tasks

- [Opening views](#)
- [Moving and docking views](#)

Related reference

- [Project Explorer](#)

Properties View

The properties view displays property names and values for a selected item from the Project Explorer.

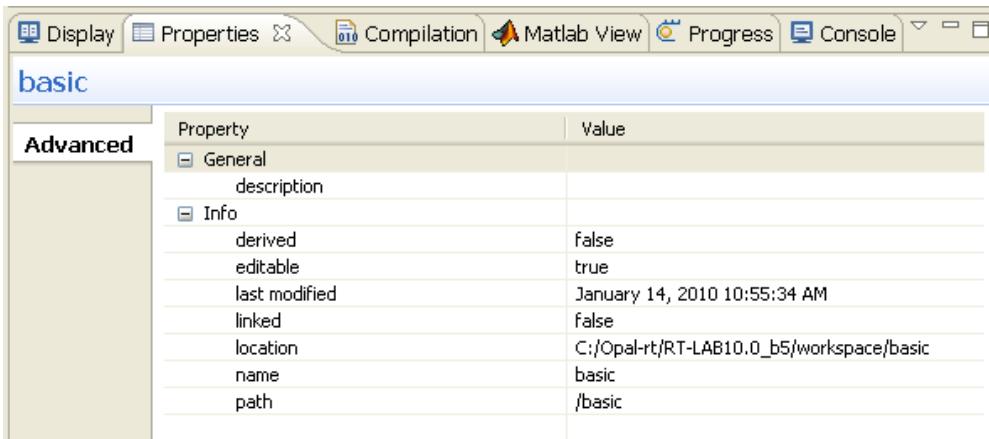
The available properties are different depending if the selected item is a project, a model, a target, or a subsystem.

When multiple items are selected, the common properties for all items are shown and the view allows the user to change the values of all items at the same time. Note however that the properties view shows the property values of the first selected item.

The following section describes the properties for each type of selected item.

Project

When a project is selected in the Project Explorer view, the properties view displays the project's properties.



The screenshot shows the MATLAB Properties View window. The title bar includes tabs for Display, Properties (which is selected), Compilation, Matlab View, Progress, and Console. The main area is titled 'basic'. On the left, there is a sidebar with 'Advanced' selected. The main table lists project properties under two categories: General and Info.

Property	Value
description	
derived	false
editable	true
last modified	January 14, 2010 10:55:34 AM
linked	false
location	C:/Opal-rt/RT-LAB10.0_b5/workspace/basic
name	basic
path	/basic

Here are the descriptions of the project properties:

description: User's description set at the project's creation.

derived: Indicates if the project is a derived resource or not. A derived resource is resource that is not original data and can be recreated from its source file.

editable: Indicates if the project is Write protected or not.

last modified: Date of the last modification of the model.

linked: Indicates if the project is a linked resource or not. A linked file or directory is a file that is stored in locations in the file system outside of the project's location.

location: Directory where the project is located on the host target.

name: Name of the project.

path: relative path of the project.

Model

When a model is selected in the Project Explorer view, the properties view displays the model's properties.

These properties are grouped into six major categories:

- Execution

- Simulation tools
- Diagnostic
- Development
- Hardware
- Info

Resource	Property	Value
	1- Execution	
	▷ environment variables	
	▷ files	
	▷ performance	
	▷ real-time	
	pause time	Infinity
	real-time communication link	UDP/IP
	real-time simulation mode	Software synchronized
	stop time	Infinity
	target platform	Windows XP/Vista/7
	time factor	1.0
	use restricted mode	false
	use restricted mode at	10
	2- Simulation tools	
	▷ Console	
	▷ Matlab	
	▷ RTW	
	3- Diagnostic	
	▷ debugging	
	▷ monitoring	
	4- Development	
	▷ Compiler	
	▷ Linker	
	5- Hardware	
	▷ Monitoring	
	▷ Performance	
	6- Info	
	derived	false
	description	
	editable	true
	last modified	
	linked	false
	location	C:/Work/realtime/Simulink/rtdemo1.mdl
	name	rtdemo1.mdl
	path	C:/Work/realtime/Simulink/rtdemo1.mdl
	size	27044 bytes

Here are the descriptions of the model properties:

Execution / environnement variables / variables: list of environment variables.

User environment variables are used to enable some settings for the target simulation without having to rebuilt the model. Most of these settings are used for debug purposes, or as workarounds for specific simulation situations.

Execution / files / extra files: list of extra files

This list specifies one or more files to be copied and transferred to (or from) the target environment.

Execution / files / file retrieval build tree: Specifies if intermediate folders are generated or not in the retrieve directory.

Execution / files / file retrieval enabled: When this option is enabled, RT-LAB retrieves files generated by the model simulation on the target nodes and transfers them to the host computer.

Execution / files / file retrieval root directory: Directory where the files retrieved by RT-LAB after the model execution will be copied.

Execution / performance / action after N overruns: Performs the action specified in the "Action to perform on overruns" field when the number of overruns detected by RT-LAB reaches this number. (Only applies when "Action to perform on overruns" is set to Reset or Pause.)

Execution / performance / action on overruns: Type of action to perform when overruns are detected. Here are the available types:

- Continue: No action is performed; Number of overruns will simply be displayed during the next pause/reset of model.
- Reset: A reset of the model is performed when the number of overruns reaches the value of the "Perform action after N overruns" field.
- Pause: A pause of the model is executed when the number of overruns reaches the value of the "Perform action after N overruns" field.

Execution / performance / detect overruns: (Only applies to models ran in hardware/software synchronized modes.) When this parameter is not checked, RT-LAB does not detect overruns. When this parameter is checked, RT-LAB detects overruns and performs the action specified in the "Action to perform on overruns" field. This property has no effect in simulation mode.

Execution / performance / number of steps without overruns: Prevents RT-LAB to detect overruns during the first steps of simulation.

Execution / real-time / pause time: This value (in seconds) specifies when the model will pause. If value is "Infinity", the model will execute until a reset is done except if a stop time is specified.

Execution / real-time / communication link: Sets the target cluster's communication medium. Here are the available choices for each target platform.

- XP/Vista/7 system: UDP/IP.
- QNX 6.x: OHCI, UDP/IP.
- RedHat: OHCI, UDP/IP, SCI, INFINIBAND.

Execution / real-time / simulation mode: Enables the user to define how the model's simulation can be executed on a QNX6, RedHat, or Windows XP/Vista/7 system.

- Simulation: in this mode, the model is not synchronized; the model starts a new computation step as soon as the previous one is completed (the model runs as fast as possible).
- Simulation with low priority: this simulation mode can be used to run a simulation in the background while working with other applications on the same computer.

Note: Please note that the Simulation with low priority mode is only available when working with a Windows target.

- Software Synchronized: in this mode, real-time synchronization of the entire simulation is achieved by the OS, using the CPU clock as a reference. Depending on the resolution available on the OS, some sampling times may not be obtained using this mode (e.g. on QNX6.1 the smallest sampling time available is 500 µs).
- Hardware Synchronized: in this mode, an I/O board clock is used to synchronize the entire simulation.

Execution / real-time / stop time: This value (in seconds) specifies when the model will stop. If value is "Infinity", the model will execute forever. You don't need to recompile the model to apply a new value.

Execution / real-time / target platform: Allows the user to choose the target platform on which to run the simulation (QNX6, Redhat, Redhawk, Windows XP/Vista/7 system).

Execution / real-time / time factor: This value, when multiplied by the model's basic calculation step (or fixed step size), supplies the final calculation step for the system. The time factor can only be changed when the simulation execution is paused and is only available in Synchronized (hardware/software) mode.

Note: This parameter enables you to change the I/O acquisition speed. Because the model always uses the basic calculation step, changing the time factor for the I/O may give results which differ from the offline simulation. This parameter should be used only to determine the simulation's minimum calculation step. Once this step is determined, this parameter must be brought back to a value of 1. The model's basic calculation step must be updated accordingly by setting the correct value in the Simulink/SystemBuild model and by recompiling it.

Execution / real-time / use restricted mode (For advanced user only): This setting can be used only when the model's current target platform is QNX 6.x. When this platform is selected, a given model can be loaded either in 'Free Clock' mode or in 'Restricted Clock' mode.

In Free-Clock mode, the user can load one non-Software Synchronized model on a given target node and is free to use any time-step value (within the boundaries of the QNX 6.x restrictions). If the user tries to load another non-Software Synchronized model on the same target node, it will be stopped with a relevant error message.

In Restricted-Clock mode, the user can load a series of non-Software Synchronized models on a given node. In this mode, each model time-step must be consistent (i.e. a multiple of) with a specific base Clock Period that should be specified by the user. Each time an additional model is being loaded on this specific target, its time-step will be checked against the current base Clock Period to validate the load operation.

Execution / real-time / use restricted mode at: (For advanced user only): Defines the Clock Period in microseconds specified by the user for the Use Restricted mode option. (Only available when the Use Restricted mode option is enabled).

Simulation tools / Console / handle console: Enables the console to be automatically opened and closed, when a model is Loaded or Reset. The console starts automatically at the first model's execution.

Simulation tools / Matlab / command after opening model: Matlab command to execute after opening the model. For example, MyInitFile.m, where MyInitFile includes commands that initialize model variables.

Simulation tools / Matlab / command before opening model: Matlab command to execute before opening the model. For example. MyInitFile.m, where MyInitFile includes commands that initialize model variables.

Simulation tools / Matlab / step size: Displays the basic calculation step of the model as specified in the Simulation>Configuration Parameters>Solver menu of the Simulink model file.

Simulation tools / RTW / user TLC file: Specifies which tlc file to use during the compilation. When Automatic is selected, the tlc file corresponding to MATLAB version is automatically selected. This option is available for SIMULINK models only.

Simulation tools / RTW / user TMF file: Specifies which template makefile to use during the compilation. When Automatic is selected, the template makefile corresponding to MATLAB version is automatically selected. This option is available for SIMULINK models only.

Diagnostic / debugging / compile in debug: When this option is enabled, the compiler's optimization options are removed and replaced by debug options during model compilation. This option is useful when debugging a model.

Diagnostic / debugging / enable watchdog: Enables or disables the Watchdog functionality. The Watchdog ensures that all nodes are running normally. There are as many watchdogs as there are computation subsystems in the model. At regular user-defined time intervals, the watchdog verifies that the model is still running and that processes are still active. If an error occurs, the watchdog stops the simulation. This option is not available on Windows target.

Diagnostic / debugging / extended timeout: All timeouts are extended when this option is enabled. It is typically used when debugging the model.

Diagnostic / debugging / watchdog timeout: Enables or disables the Watchdog functionality. The Watchdog ensures that all nodes are running normally. There are as many watchdogs as there are computation subsystems in the model. At regular user-defined time intervals, the watchdog verifies that the model is still running and that processes are still active. If an error occurs, the watchdog stops the simulation. This option is not available on Windows target.

Diagnostic / monitoring / enable monitoring: Enables or disables the monitoring during the execution of the model.

Diagnostic / monitoring / message precision factor: (For advanced user only): Defines the precision of the timer reporting the time required to perform one calculation step in microseconds. The default value is 0 (no printout). Value=1 prints the step size of the model every 1,000,000 steps with a precision of 1 μ s. Value=100 prints the model's step size every 10,000 steps with a precision of 100 μ s and so on.

Warning: This option may disturb the real-time simulation performances.

Diagnostic / monitoring / target display information: Represents the amount of information that is displayed in the display window during load and execution. Setting the value from MINIMAL to EXHAUSTIVE allows the user to retrieve more information about communication and I/Os initialization for example. This can be used to debug unexpected model behaviors.

Note: Please note that Exhaustive mode must only be used for debug purposes as it can influence the real-time performances.

Development / Compiler / user compilation command: User command to be called on the target during the compilation process.

Development / Compiler / user compilation options: Options used by the target compiler during the compilation process. Refer to the target compiler documentation for specific information on the possible compiler options (e.g. -xxx).

Development / Compiler / user include path: Target path(s) where user include files can be found. Used to search for include files required to compile user-written code, when the include files do not reside in already-searched paths.

Development / Compiler / user source files: Names of source files to be compiled in addition to the code generated by RTW/AutoCode. Used to incorporate user-written code into the model executable (e.g. file1.c file2.c file3.c).

Development / Linker / user external libraries: Names of user-specified libraries to be used when incorporating user-written code, in addition to the standard system libraries (Matlab and RT-LAB).

Development / Linker / user library path: Target path(s) where user libraries can be found. Used to search for libraries other than the MATLAB, MATRIXX and RT-LAB libraries when incorporating user-written code.

Development / Linker / user linker options: Options used by the target linker during the linkage process. Refer to the target linker documentation for specific information on the possible linker options (e.g. -xxx).

Hardware / Monitoring / Abort compilation when bitstream missing: When this option is enabled, if some bitstreams required by a model are not present in the model directory, RT-LAB will stop the compilation and generate an error. RT-LAB will generate only a warning if this setting is disabled.

Hardware / Monitoring / reset model on IOs missing: When this option is checked and the model is loaded, RT-LAB will stop loading and will generate an error, if some I/O boards required by the model are not detected on the target(s). RT-LAB will generate only a warning if this setting is unchecked. Note that only a few I/O boards (mainly the Opal-RT TestDrive boards) support this option. Most I/O blocks force a reset of the model if the boards are not detected at load time.

Hardware / Performance / cacheable DMA memory access: When this option is checked, RT-LAB will enable the cache of the memory areas used by DMA buffers yielding to faster transfer rates and increased model performances. However, this option should be used with caution since it can affect proper reading of I/O data using DMA buffers. This option is available on QNX6 only.

Info / derived: Indicates if the model is a derived resource or not. (a derived resource is resource that is not original data and can be recreated from its source file)

Info / editable: Indicates if the model is Write protected or not.

Info / last modified: Date of the last modification of the model.

Info / linked: Indicates if the model is a linked resource or not (a linked file is a file that is stored in locations in the file system outside of the project's location).

Info / location: Directory where the model is located on the host target.

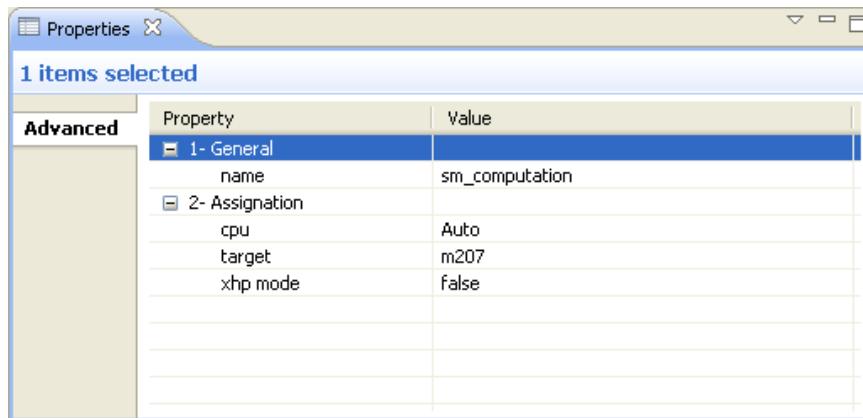
Info / name: Name of the model.

Info / path: Path where the model is located on the host target.

Info / size: size in bytes of the model's file.

Real-time subsystem

When a real-time subsystem, such as master and slave subsystem, is selected in the Project Explorer view, the properties view displays the subsystem's properties.



Here are the descriptions of the real-time subsystem's properties:

General / name: Name of the real-time subsystem.

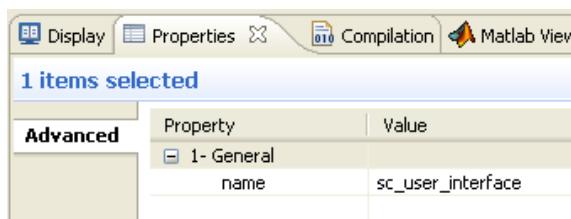
Assignton / target : Name of the target where the subsystem will be executed.

Assignton / cpu : (For advanced users only). CPU cores allocated (and optionally specified) for the subsystem and its worker threads. (available only on QNX 6 and Red Hat Linux)

Assignton / XHP mode : If checked, the subsystem will be executed in eXtreme High Performance (XHP) mode (available only on QNX 6 and RedHat Linux).

Console subsystem

When a console subsystem is selected in the Project Explorer view, the properties view displays the console's properties.



Here are the descriptions of the console properties:

General / name: Name of the console subsystem.

Target

When a target is selected in the Project Explorer view, the properties view displays the target's properties such as the IP address, the detected operating system and the number of CPUs.

Property	Value
1- Overview	
IP address	192.168.0.139
name	redhat_i7
2- Operating System	
detected platform	Redhat
version	2.6.29.6-opalrt-2
3- Hardware	
architecture	i686
CPU speed [MHz]	3200
disk space [MO]	124597
number of CPUs	8

Here are the descriptions of the target properties:

IP address: IP address of the target.

name: User name set at the target creation.

detected platform: Operating system detected Windows XP/Vista or Redhat or QNX.

version: Version of the operating system detected.

architecture: Architecture type of the target (for instance x86pc or x86).

CPU speed: Speed in MHz of the target CPU.

disk space: Free disk space in MO of the hard disk.

number of CPUs: number of CPUs of the target.

Contextual Menu

The properties view provides a contextual menu when right-clicking on properties. The following figures shows the contextual menu.

Property	Value
1- Execution	
+ environment variables	
+ files	
performance	
action after N overruns	10
action on overruns	Copy
detect overruns	true
number of steps without ov	10
	Restore Default Value

The available menu items are:

Copy:

This command is used to copy the Property and the current value of the selected item. The user can then paste the selection in external application.

Restore Default Value:

This command is used to restore the default value defined in the preference pages.

Matlab View

The **Matlab View** provides a quick access to the local Matlab application inside RT-LAB. From here, it is possible to make modification to the offline model with Matlab.

The **Matlab View** appears in a tabbed notebook:

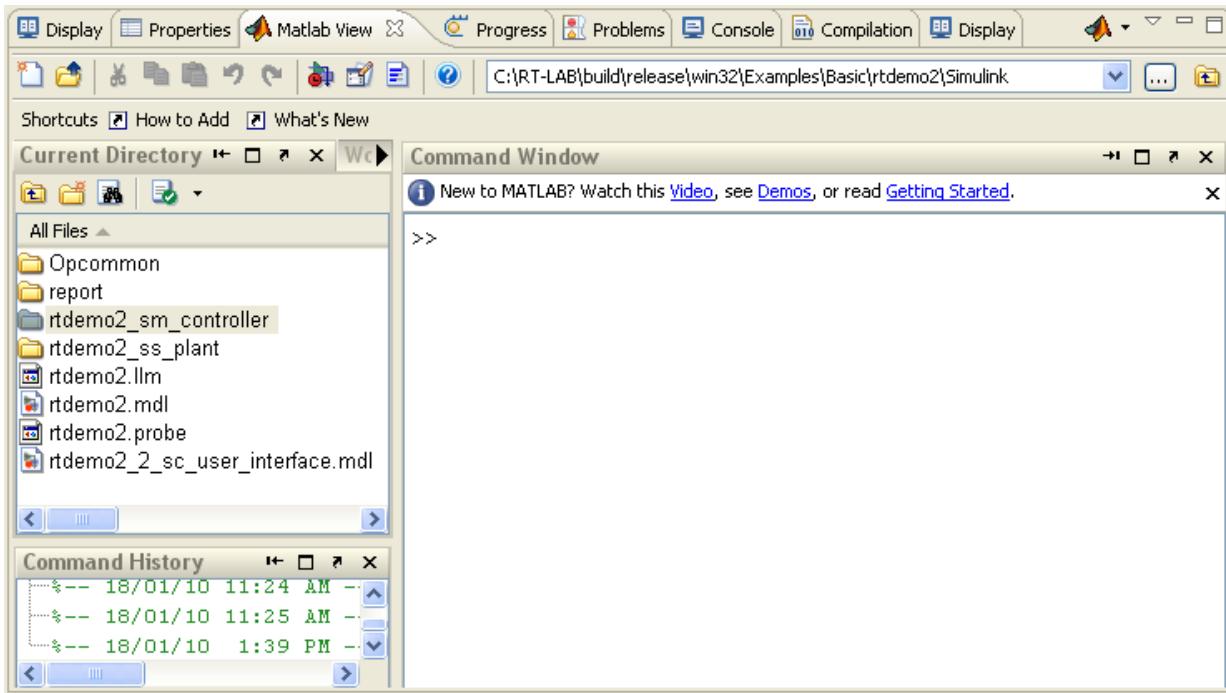


Figure 29:Matlab View

The next section presents how to open Matlab. Other sections show some examples and explain the toolbar.

By default, the **Matlab View** is included in the Edition perspective. To add it to the current perspective, click Window > Show view > Other... > RT-LAB > Matlab View

How to open Matlab within RT-LAB

There are two ways to open Matlab inside the **Matlab View**. The first one is to use the toolbar menu button to open a specified version. The second one is to click on the Open Matlab button.

Open Matlab with toolbar menu button

Inside the **Matlab View**, you can click on the Matlab button and open a specified version of Matlab. The specified version of Matlab will open inside the Matlab view.

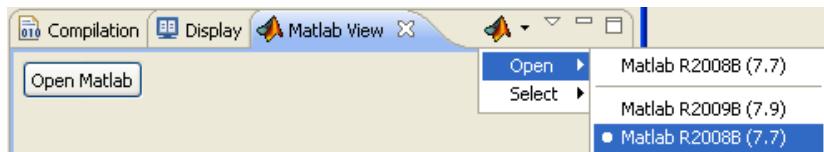


Figure 30:Open Matlab with toolbar

Open Matlab using the default matlab version

Click on the Open Matlab button, located inside the **Matlab View**.



Figure 31:Open Matlab with button

What does the Matlab View offer

One of the coolest things with this feature is that you don't have to switch between Matlab and RT-LAB to use Matlab functionalities.

The embedded Matlab View gives opportunity to:

- Edit or modify the model before the compilation process. For more information, see [Building models](#).
- Start an offline Matlab simulation
- Change parameters before the compilation process
- Execute scripts to analyze data
- Use other Matlab functionalities...

Note: However, because there is no Matlab environment on the real-time operating system, any changes to the Simulink model requires that you recompile it to affect the real-time simulator.

Disabled MATLAB Versions

On Windows operating system, the selection menus for the Matlab versions display all Matlab version supported by RT-LAB. However, RT-LAB disables the Matlab versions that has not been installed or that has not been registered in Windows COM interfaces.

To register a Matlab version in Windows COM interfaces:

- open the command prompt of the operating system,
- set the working directory to the Matlab installation directory and then
- execute the following command
matlab.exe /regserver

This command will enable the specific Matlab version so it can be used in RT-LAB. Then, to verify that Matlab has been registered correctly, perform the following steps:

- execute the following command
Matlab.exe /automation
- If Matlab doesn't start correctly, Matlab must be reinstalled using the normal installation procedure.

If none of the steps above solved your problem, please contact RT-LAB support.

Toolbar

The toolbar in the **Matlab View** contains the following buttons:

ICON	DESCRIPTION
------	-------------

**Open Matlab**

Opens the selected Matlab version. It is also possible to open Matlab using the default Matlab version.

**View Menu**

Opens the View Menu.

View menu

The menu in the **Matlab View** contains the following items:

Close Matlab

This command closes the instance of Matlab in the current view.

Close View

This command closes the **Matlab View** but doesn't close the Matlab instance.

Close Matlab and View

This command closes the instance of Matlab and the **Matlab View**.

Related concepts

[Views](#)
[Perspectives](#)
[Toolbars](#)

Related tasks

[Opening views](#)
[Moving and docking views](#)

Related reference

Console View

The **Console View** is a generic view that may contain different Console types. A Console basically displays formatted text. Some Consoles enable user input, for example the **Interactive Python Console**.

By default, the RT-Lab Workbench includes three different Consoles:

- An **Interactive Python Console** for Python development:

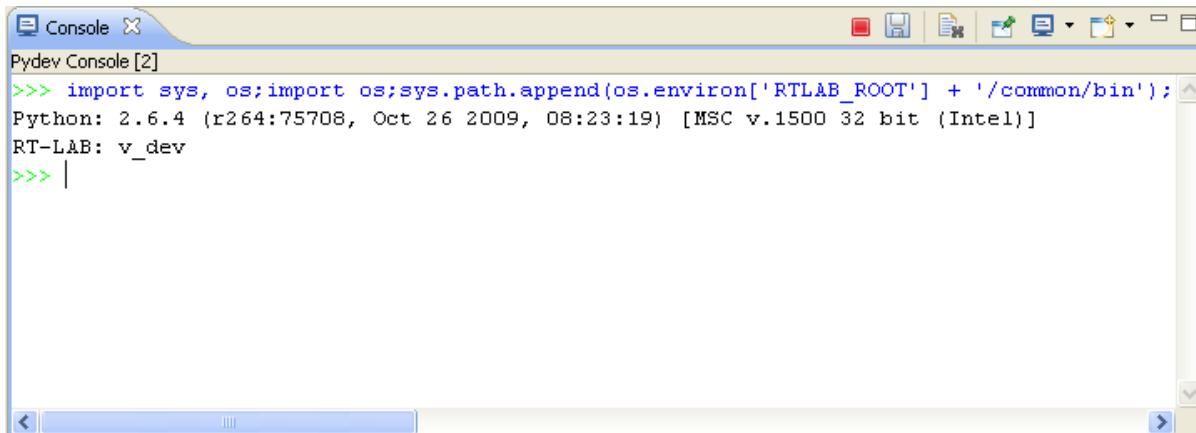


Figure 32:Interactive Python Console overview

- A **Target Console** that displays output from target nodes when executing remote commands:

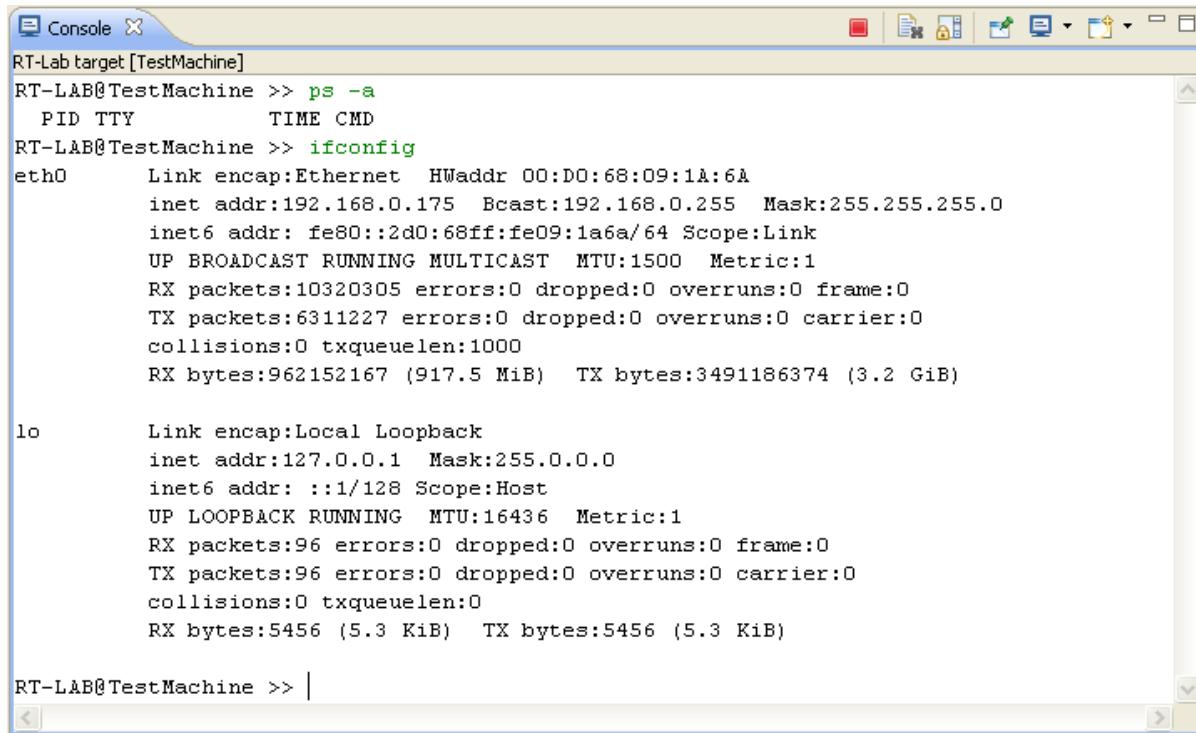


Figure 33:Target Console overview

- A **Log Console** that provides internal debug information. This Console can only be used if the "Advanced User" **Capabilities preference page** is enabled:

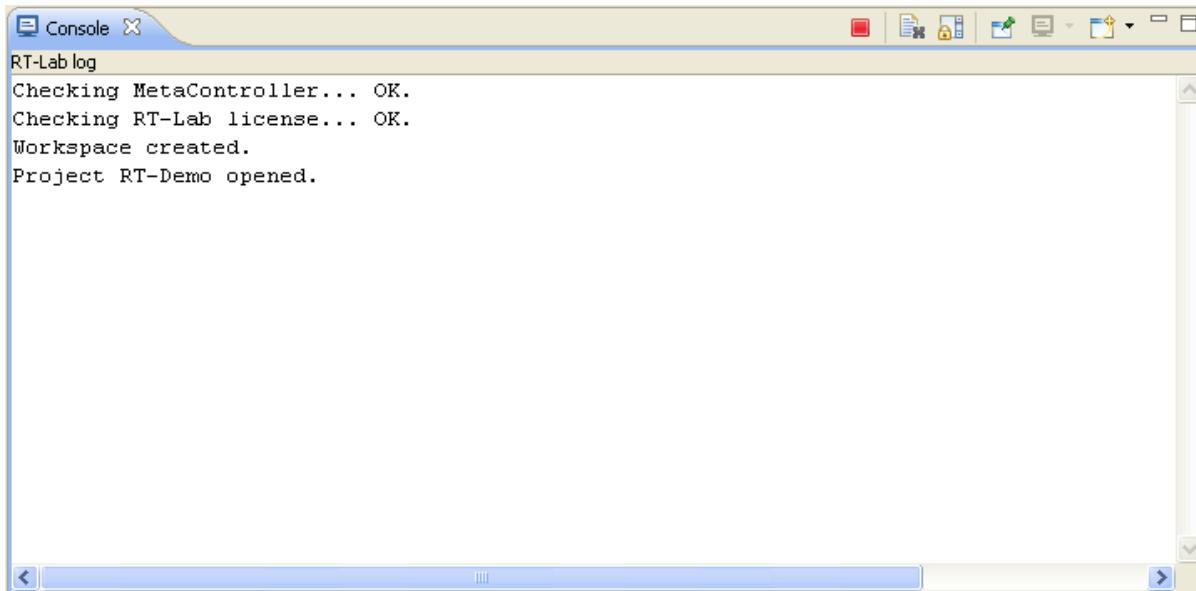


Figure 34:Log Console overview

Several Console Views may be opened at the same time in the workbench, but they all share the same list of Consoles. To add a Console View to the current perspective, click **Window > Show view > Other... > RT-LAB > Console** or use the Open Console button of an existing Console View (See below).

This is useful when you want to display several Consoles side-by-side, for example to compare two command outputs.

Toolbar

The toolbar of the **Console View** provide some commands that are common to all Console types:

Command	Name	Description	Availability
	Clear Console	Clears the current console.	Toolbar and Context menu
	Scroll Lock	Toggles the Scroll Lock	Toolbar
	Pin Console	Pins the current console to remain on top of all other consoles in the current Console View.	Toolbar
	Display Selected Console	Opens a listing of current consoles and allows you to select which one you would like to see.	Toolbar
	Open Console	Opens a new console of the selected type.	Toolbar

Most Console types provide the following additional commands:

Command	Name	Description	Availability
	Close Console	Closes the current console.	Toolbar
	Save Console session	Saves the current content of the Console to a file.	Toolbar

Interactive Python Console

The **Interactive Python Console** is shown in the **Console View**. This Console is provided by the PyDev plugin, available at <http://pydev.org>

Here is what the **Interactive Python Console** looks like:

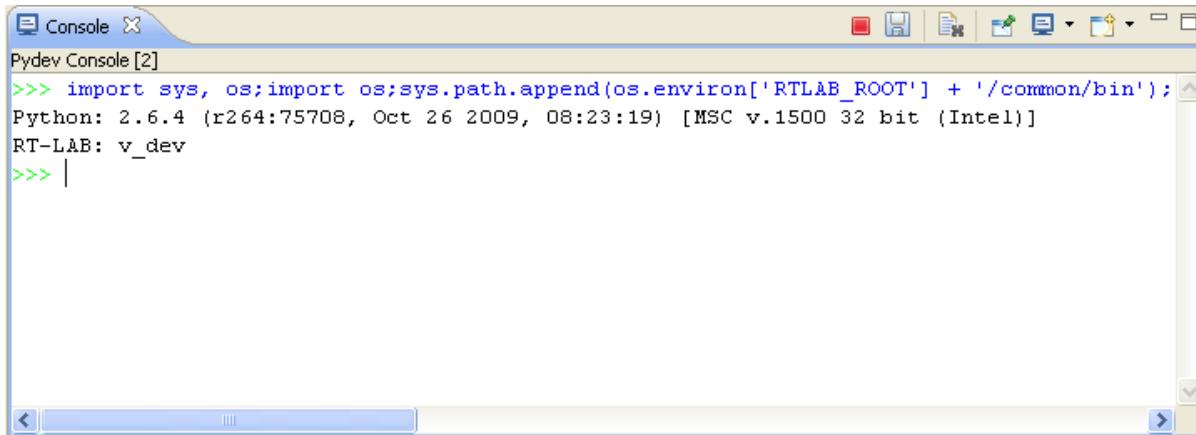


Figure 35:Interactive Python Console: Overview

To open a new **Interactive Python Console**, you can use the main application toolbar:

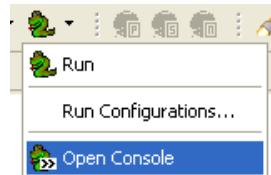


Figure 36:Python menu

or the corresponding menu: **Tools > Python > Open Console**.

A dialog is displayed that allows you to choose among available console types:



Figure 37:Python Console dialog

Repeat these operations to open several Consoles in the same View. These Consoles may be of different types. Use the toolbar to switch between opened Consoles.

Once started, the **Interactive Python Console** is able to launch Python scripts and to execute Python commands. This is particularly useful when working with RT-Lab projects, since RT-Lab provides a complete Python module. This module, named **RtlabApi**, allows you to control any part of a simulator, from configuration to execution. All available functions are described in the Python API Reference Guide.

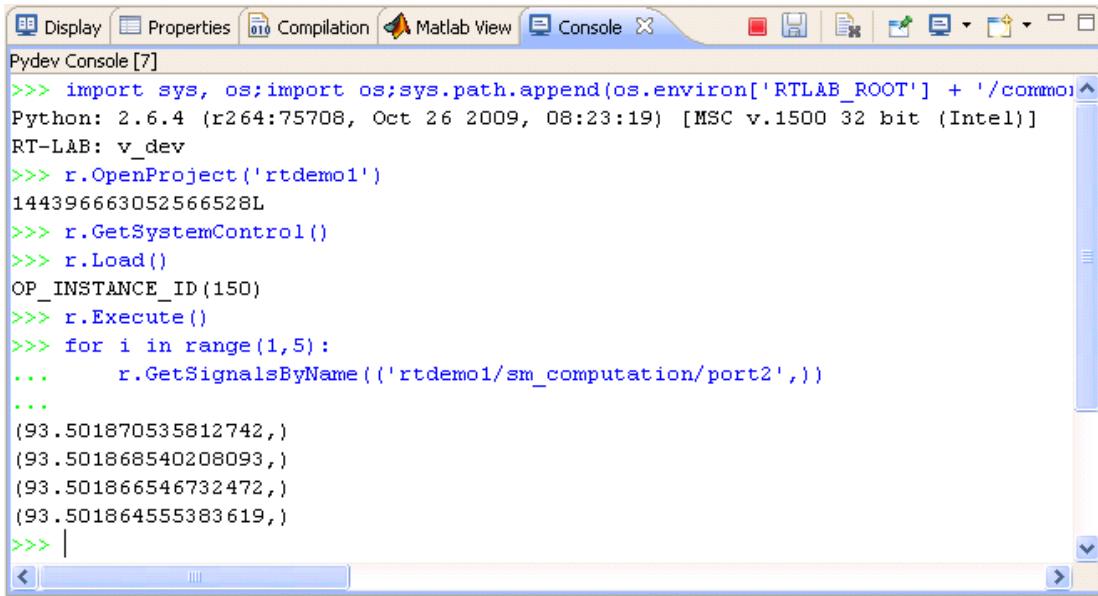
In order to use this module, it is necessary to import it with the following command:

```
>>> import RtlabApi
```

However, by default, when opening an **Interactive Python Console**, some modules are automatically imported, including the RT-Lab API. The RT-Lab API is also given the alias 'r'. Use the PyDev preference page to edit these initial commands (see [Configuration](#)).

Using the RT-Lab API through the **Interactive Python Console** is a good way to test some commands that would be later integrated to a Python script and thus create a full automated sequence.

Here is an example of what the **Interactive Python Console** can do:



The screenshot shows a Pydev Console window with the title 'Pydev Console[7]'. The console displays the following Python code and its output:

```

>>> import sys, os;import os;sys.path.append(os.environ['RTLAB_ROOT'] + '/common')
Python: 2.6.4 (r264:75708, Oct 26 2009, 08:23:19) [MSC v.1500 32 bit (Intel)]
RT-LAB: v_dev
>>> r.OpenProject('rtdemo1')
144396663052566528L
>>> r.GetSystemControl()
>>> r.Load()
OP_INSTANCE_ID(150)
>>> r.Execute()
>>> for i in range(1,5):
...     r.GetSignalsByName(('rtdemo1/sm_computation/port2',))
...
(93.501870535812742,)
(93.501868540208093,)
(93.501866546732472,)
(93.501864555383619,)
>>> |

```

Toolbar

The toolbar and the context menu provide some commands to control the **Interactive Python Console**:

Command	Name	Description	Availability
	Terminate	Closes the current console.	Toolbar
	Save	Saves current content to a file.	Toolbar
	Clear Console	Clears the current console.	Toolbar and Context menu
	Pin Console	Pins the current console to remain on top of all other consoles in the current Console View.	Toolbar

Command	Name	Description	Availability
	Display Selected Console	Opens a listing of current consoles and allows you to select which one you would like to see.	Toolbar
	Open Console	Opens a new console of the selected type.	Toolbar

Shortcuts

When typing commands in the [Interactive Python Console](#), the following shortcuts are available:

SHORTCUT	ACTION
Ctrl + Space	Code completion.
PageUp	Displays history in a dialog.
Up / Down Arrows	Cycles through the history.
Esc	Clears current line.
Ctrl + f	Opens the search dialog.

Configuration

Enabling the *Advanced User Capabilities preference page* allows you to configure the [Interactive Python Console](#).

Go to **Window > Preferences > PyDev > Interactive Console**:

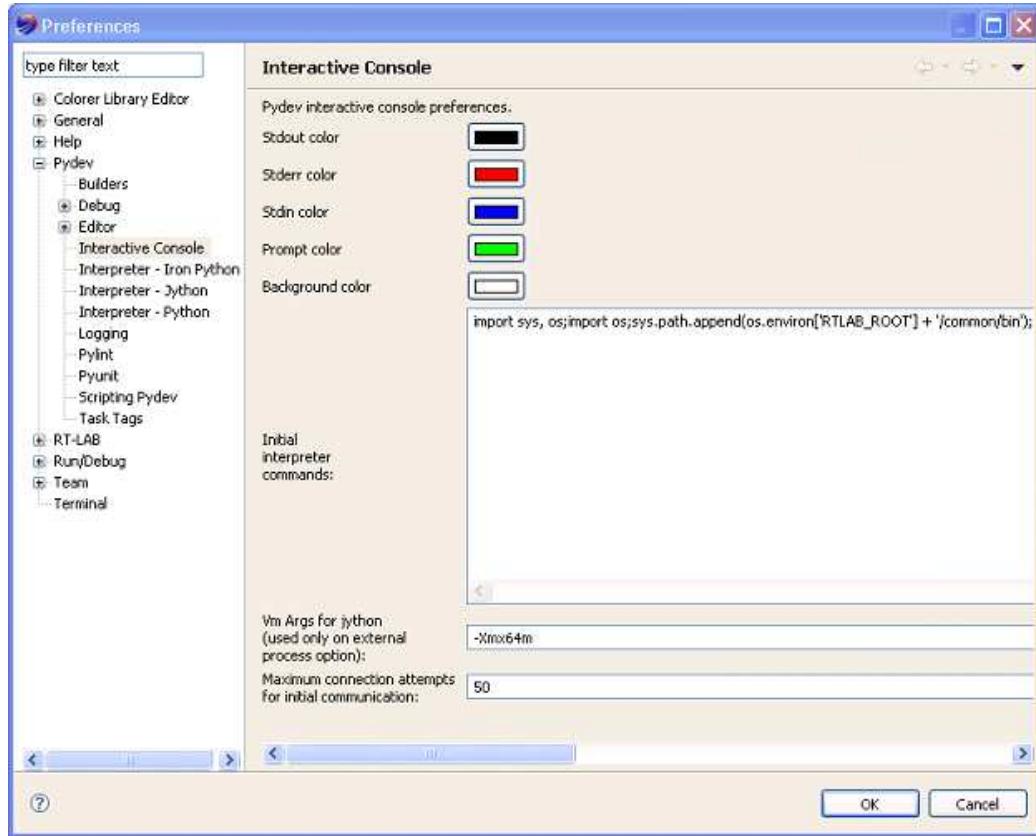


Figure 38:Python Console settings

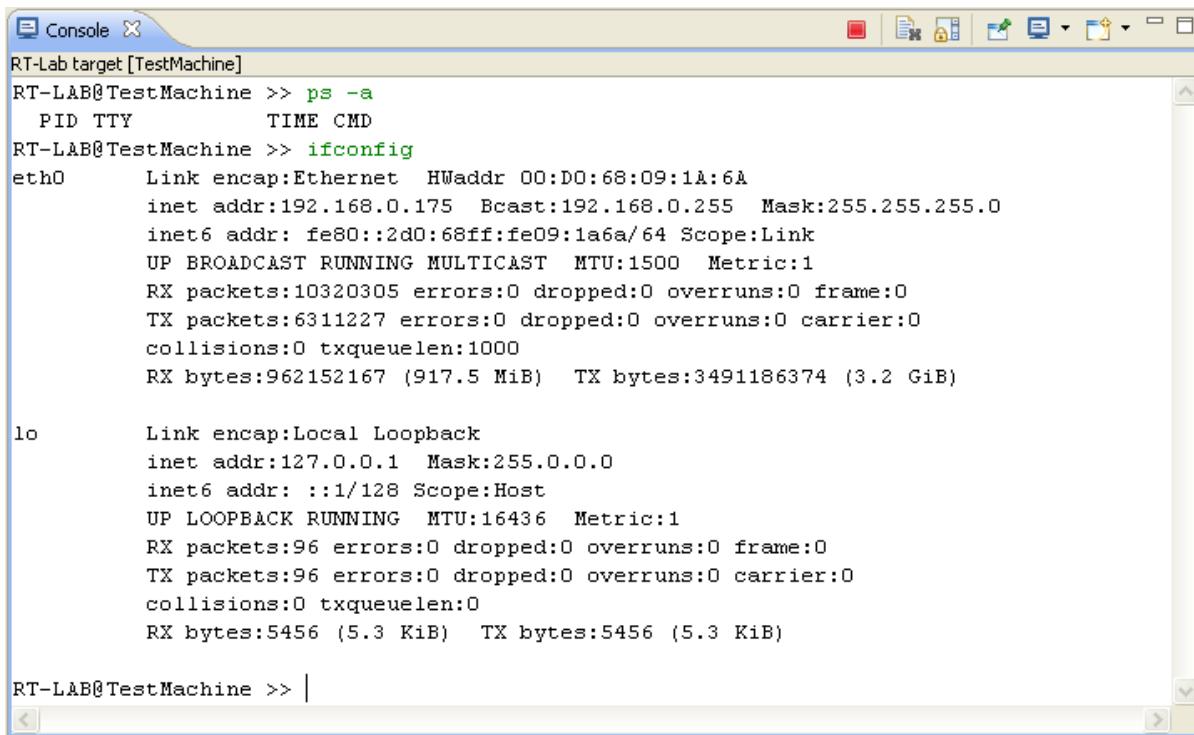
You can also change the current Python version and settings in **Window > Preferences > PyDev > Interpreter - Python**.

References

You can find additional information on the official PyDev web site: <http://pydev.org/>

Target Console

The **Target Console** is shown in the **Console View**. It is a simple Shell interface that looks similar to a **Terminal View**:



```

Console [RT-Lab target [TestMachine]]
RT-LAB@TestMachine >> ps -a
  PID TTY      TIME CMD
RT-LAB@TestMachine >> ifconfig
eth0      Link encap:Ethernet HWaddr 00:D0:68:09:1A:6A
          inet addr:192.168.0.175 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::2d0:68ff:fe09:1a6a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:10320305 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6311227 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:962152167 (917.5 MiB) TX bytes:3491186374 (3.2 GiB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:96 errors:0 dropped:0 overruns:0 frame:0
          TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:5456 (5.3 KiB) TX bytes:5456 (5.3 KiB)

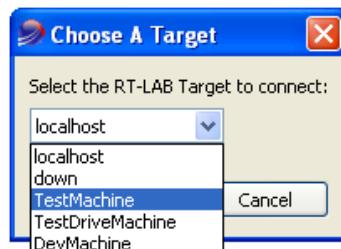
RT-LAB@TestMachine >> |

```

Figure 39:Target Console: Overview

The **Target Console** automatically opens when you execute a command or launch a Python script on a Target from its context menu.

If the Advanced **Capabilities preference page** is enabled, you can open a new **Target Console** from the Console View Menu. A dialog is displayed to select the Target to connect to this new Console:



Toolbar

The toolbar and the context menu provide some commands to control the **Target Console**:

Command	Name	Description	Availability
	Terminate	Closes the current console.	Toolbar

Command	Name	Description	Availability
	Clear Console	Clears the current console.	Toolbar and Context menu
	Scroll Lock	Toggles the Scroll Lock	Toolbar
	Pin Console	Pins the current console to remain on top of all other consoles in the current Console View.	Toolbar
	Display Selected Console	Opens a listing of current consoles and allows you to select which one you would like to see.	Toolbar
	Open Console	Opens a new console of the selected type.	Toolbar

Log Console

The **Log Console** is shown in the **Console View**. It displays internal debug messages and needs the Advanced **Capabilities preference page** to be used.

Here is what the **Log Console** looks like:

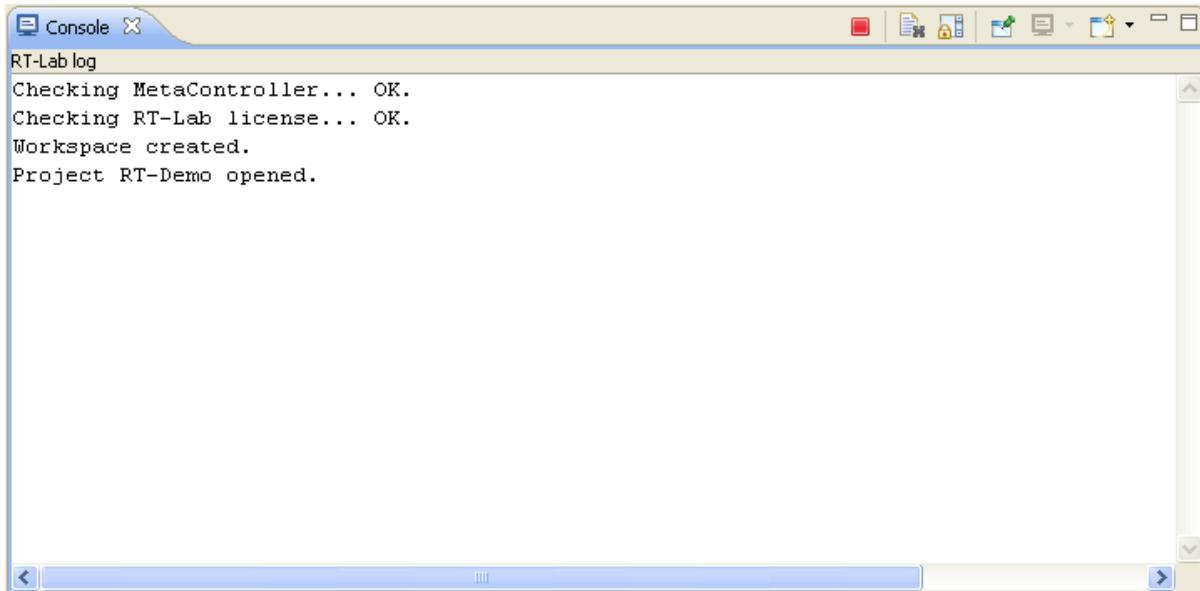


Figure 40:LogConsole: Overview

The **Log Console** can be opened using the “Open Console” command of the **Console View**.

Toolbar

The toolbar and the context menu provide some commands to control the **Log Console**:

Command	Name	Description	Availability
	Terminate	Closes the current console.	Toolbar
	Save	Saves current content to a file.	Toolbar
	Clear Console	Clears the current console.	Toolbar and Context menu
	Scroll Lock	Toggles the Scroll Lock	Toolbar
	Pin Console	Pins the current console to remain on top of all other consoles in the current Console View.	Toolbar
	Display Selected Console	Opens a listing of current consoles and allows you to select which one you would like to see.	Toolbar

Command	Name	Description	Availability
	Open Console	Opens a new console of the selected type.	Toolbar

Variables Table View

The **Variables Table View** displays one or more sets of RT-LAB variables, such as Signals and Parameters, in a tabbed notebook:

The screenshot shows the Variables Table View window titled "Demo". The table contains the following data:

Name	Path	Value	Data Type	Complex	Size	RTW	Access	
signal1	MyProject/matrixTest/sm_master/Constant/port1	1.0	Double	False	Scalar (1x1)	True	Read-Only	
[+]	Value	MyProject/matrixTest/sm_master/Constant/Value	1.0	Double	False	Scalar (1x1)	True	Read-Write
[+]	Value	MyProject/matrixTest/sm_master/Matrix/Value	<2x3 Double>	Double	False	Matrix (2x3)	True	Read-Write
[+]	X0	MyProject/matrixTest/ss_slave/Memory/X0	0.0	Double	False	Scalar (1x1)	True	Read-Write

At the bottom, there is a filter field, a search field, and status information: "4 Loaded - 4 Shown - 0 Selected - Sort: [Path (FWD)]".

Figure 41:Variables Table View, Overview

By default, the **Variables Table View** is included in the Edition perspective. To add it to the current perspective, click Window > Show view > Other... > RT-LAB > Variables Table.

Working Sets

Variables **Working Sets** are intended to represent logical groups of RT-LAB variables, for example the main parameters of a controller. This allows you to focus on a particular set of variables without having to search for them within an entire model.

Several **Variables Table Views** can be used to display different working sets. The same working set can also be shared between several Variables Tables.

Selecting Working Sets

When a new **Variables Table View** is created, it opens a default working set. This working set is fully functional but will not be saved when RT-LAB is closed. To have a working set restored the next time RT-LAB is opened, you must create a working set with a unique name.

The easiest way to do so is to click on the "New Working Set" button. Enter a name and click OK to create a new working set containing all the variables currently displayed in the **Variables Table View**. This working set is then automatically opened in the Variables Table and its name appears at the top of the table.

This method can also be used to merge several working sets into a new one: select the working sets you want to merge (see below) then create a new working set. Duplicates will automatically be removed.

To select which working sets should be shown in a Variable Table, click on the "Select Working Sets" button to open the following dialog:

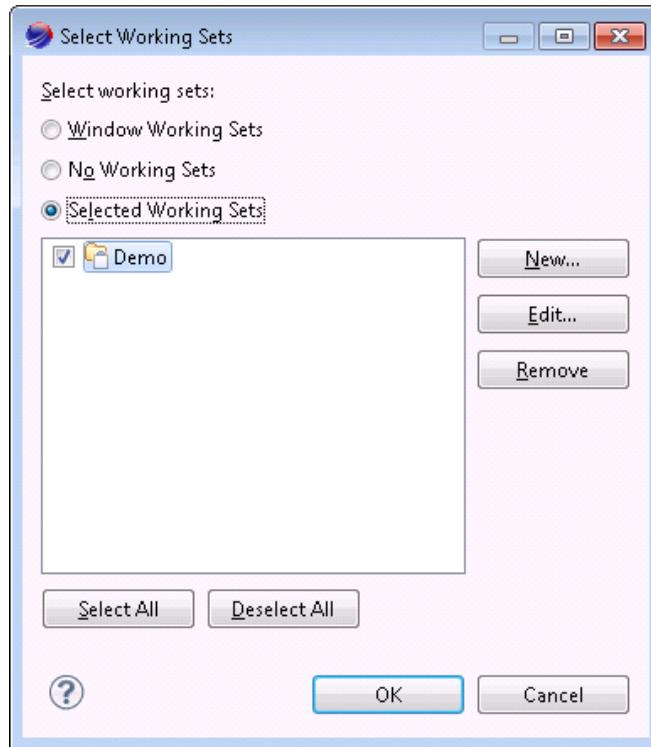


Figure 42:Select Working Sets dialog

This dialog allows you to create, edit and delete variables working sets.

Note that variables working sets are common to the entire workbench, which allows them to be used in the **Project Explorer** like other types of **Working Sets**:

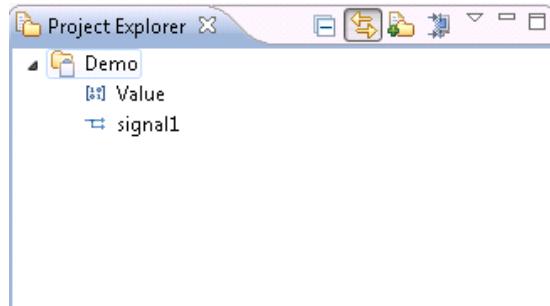


Figure 43:Using variables working sets in the Project Explorer

Editing Working Sets

In the “Select Working Sets” dialog, clicking on the “Edit...” button opens a new dialog that can be used to choose the variables that will be part of the selected working set:

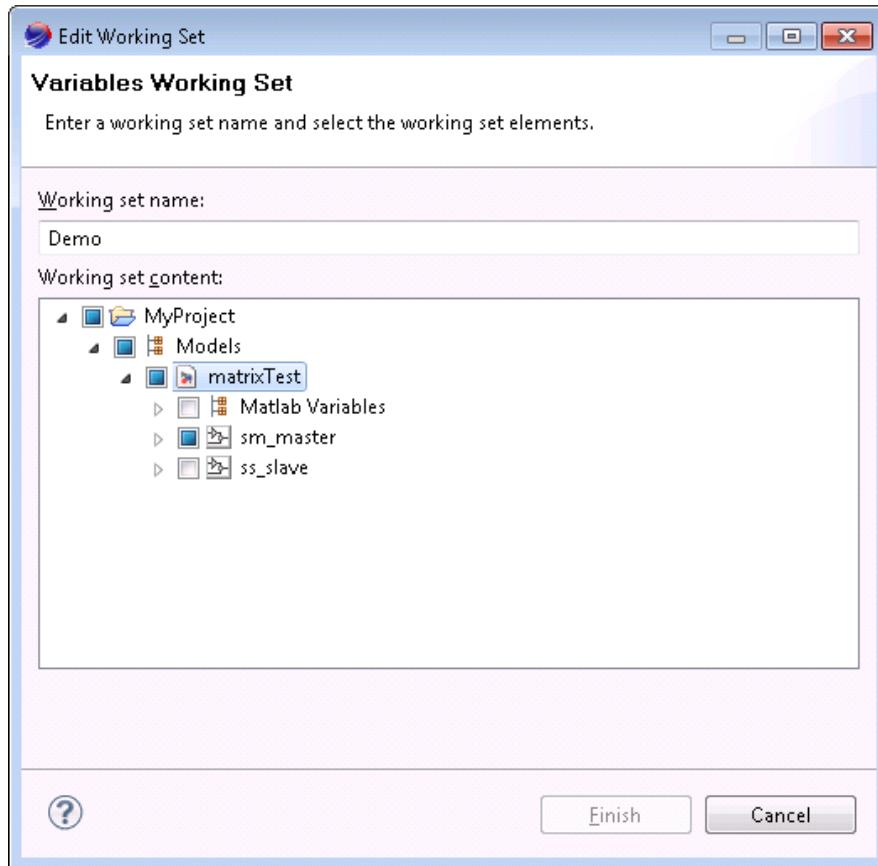


Figure 44:Edit Working Set dialog

An easier way of adding elements to a working set is to open it in a Variables Table and to drag and drop variables from the **Project Explorer**, the **Search View** or another **Variables Table View**. You can also drag a block, a subsystem or even a model to add all the contained variables to the working set.

If the Variables Table displays several working sets, the variables will be added to each of them.

It is also possible to right-click on a variable or a block (or a combination of them) in the **Project Explorer** or the **Search View** and then click on "Add to Working Sets":

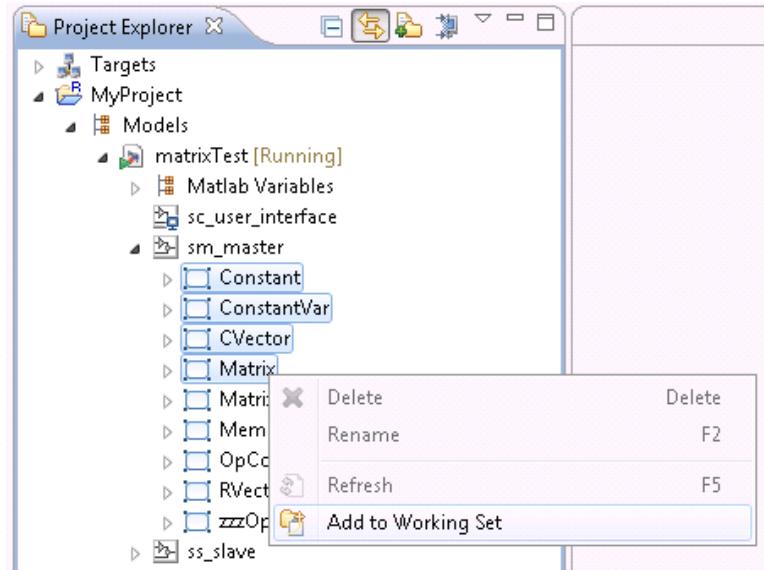


Figure 45:Context menu: Add to Working Set

This will open the "Select Working Set" dialog described above.

To remove a variable from a working set, right-click on it in the Variables Table and click on Remove, or simply press the Delete key.

Lost elements

If a variable is not available, it will be displayed with a red-cross icon as below:

Name	Path	Value	Data T...	Complex	Size	RTW	Access
✗ Value	MyProject/matrixTest/sm_master/Constant/Value	?	?	?	?	?	?
✗ port1	MyProject/matrixTest/sm_master/Constant/port1	?	?	?	?	?	?

Figure 46:Unavailable variables

This happens, for example, when the project is closed, when the model is removed from the project or when the model has been recompiled and the variable does not exist anymore.

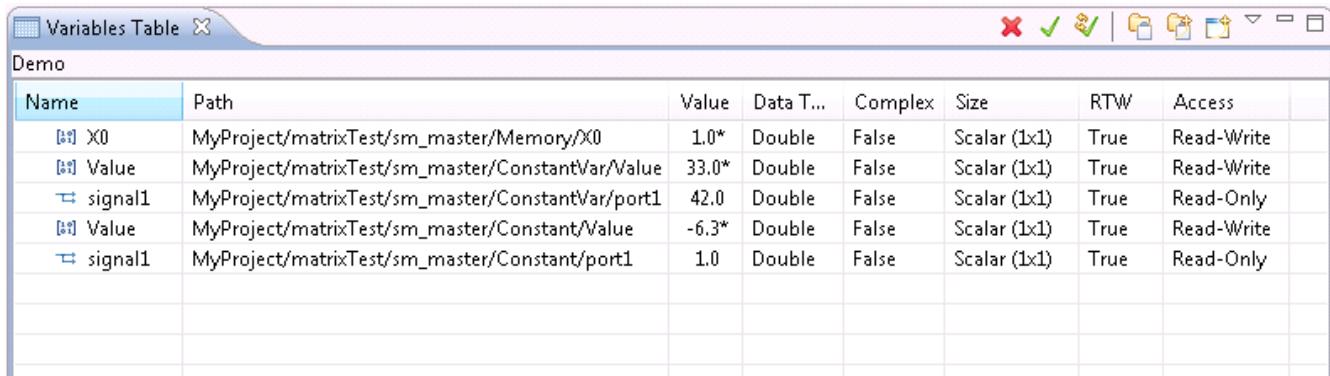
Then you can either remove the variables from the working set or fix the problem. If a variable becomes available again, the Variable Table will automatically be updated.

Editing values

When a model is running, the values of the variables are automatically updated every few seconds. You can also change the value of editable variables such as parameters.

By default, the Variables Table View is in "Auto-apply" mode. This means that as soon as you change a value and hit Enter, the value is sent to the model.

If you want to send several values at the same time (so that all the variables are updated during the same calculation step), toggle the "Auto-apply" button. Then if you modify a value and hit Enter, the value will be displayed with an asterisk indicating a non-applied change. The new value will not be sent to the model until you press the "Apply" button or you toggle the "Apply" button again. If you press the "Discard" button, all pending changes will be cancelled.



The screenshot shows the 'Variables Table' window with the title 'Demo'. The table has columns: Name, Path, Value, Data T..., Complex, Size, RTW, and Access. There are five rows of data:

Name	Path	Value	Data T...	Complex	Size	RTW	Access
X0	MyProject/matrixTest/sm_master/Memory/X0	1.0*	Double	False	Scalar (1x1)	True	Read-Write
Value	MyProject/matrixTest/sm_master/ConstantVar/Value	33.0*	Double	False	Scalar (1x1)	True	Read-Write
signal1	MyProject/matrixTest/sm_master/ConstantVar/port1	42.0	Double	False	Scalar (1x1)	True	Read-Only
Value	MyProject/matrixTest/sm_master/Constant/Value	-6.3*	Double	False	Scalar (1x1)	True	Read-Write
signal1	MyProject/matrixTest/sm_master/Constant/port1	1.0	Double	False	Scalar (1x1)	True	Read-Only

Figure 47: Variables Table with pending changes

Note that pending changes are shared between views which means that only one pending change is allowed per variable at any time.

To edit the value of a vector or a matrix, you have to use the **Variable Viewer**: double-click on the variable or right-click on it and select Show In > Variable Viewer.

Table Customization

You can sort variables by name, path, type or any property by clicking on the corresponding column header. Columns can be resized and moved by drag and drop.

The Variables Table also provides Filter and Search functionalities at the bottom of the view.

For advanced customization, open the View Menu and select Customize Table to open the following dialog:

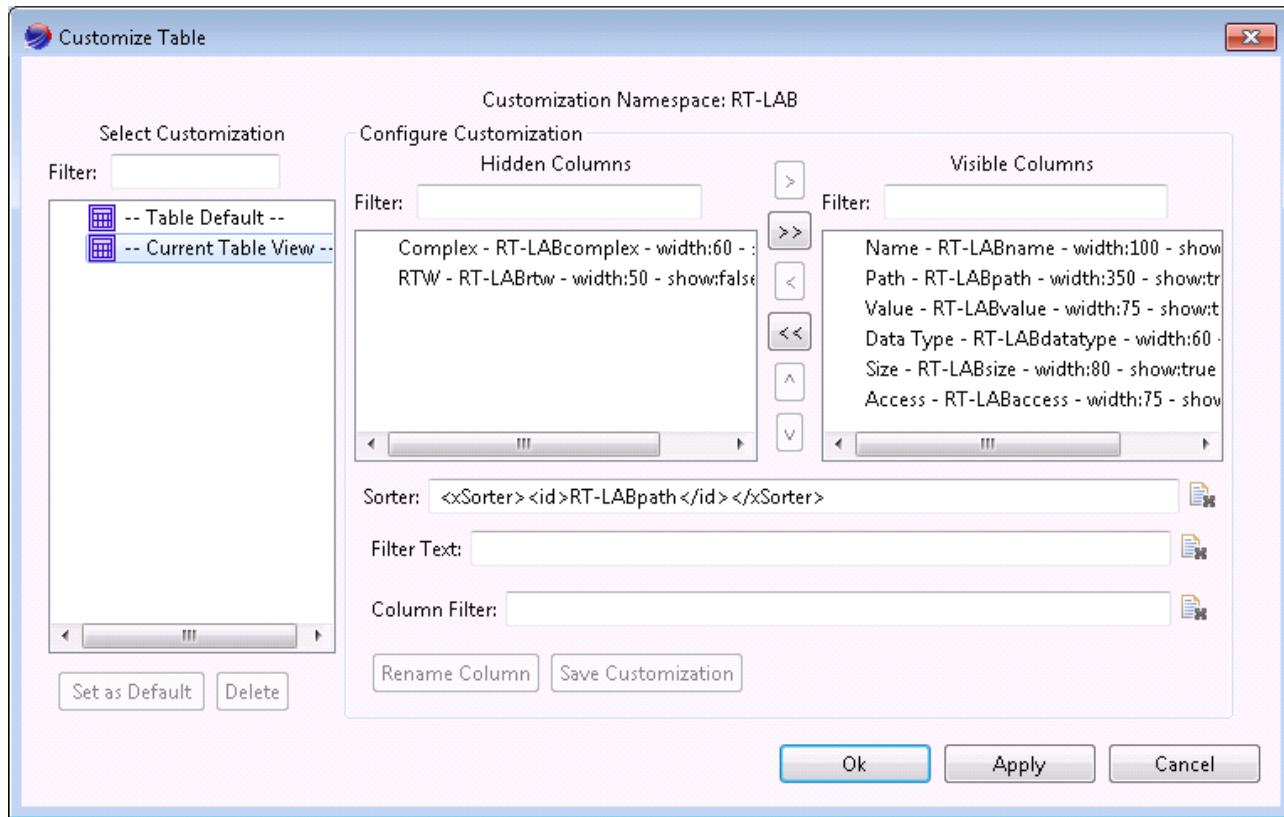


Figure 48:Customize Table dialog

Toolbar

:

Command	Name	Description
	Discard	Discards current changes.
	Apply	Applies current changes.
	Auto-apply	Toggles auto-apply mode.
	Select Working Sets	Selects the working sets to be displayed in this view.
	New Working Set	Saves current variables to a new working set.
	New View	Opens a new Variables Table view.

View Menu

Command	Description
Customize Table	Opens the configuration dialog.

Context Menu

Command	Description
Show In...	Displays the selected variable in another view.
Copy	Copies the selected lines to the clipboard.
Remove	Removes the selected variables from the current Working Sets
Select All	Selects all the variables currently displayed in the view.
Refresh	Updates the table content.

Related concepts

[Working Sets](#)
[Views](#)
[Perspectives](#)
[Toolbars](#)

Related tasks

[Opening views](#)
[Moving and docking views](#)

Related reference

[Variable Viewer](#)
[Project Explorer](#)
[Search View](#)

Probe Control Panel

The **Probe Control** panel enables you to specify the acquisition parameters for each acquisition group. The signals from a given group share the same characteristics, such as:

- size of the data frames being sent, expressed either as a number of samples per signal, or in units of time
- decimation factor
- sampling mode (repetitive or not) and re-arm delay
- blocking mode in case acquisition or transmission is interrupted
- a **write-to-file** option, that enables simulation data to be saved as a ".mat" file on the target nodes.

The Probe Control Panel can be launch using the **Tools > Probe Control** menu or using the toolbar button. Note that the panel is only available when a model is selected in the Project Explorer view.

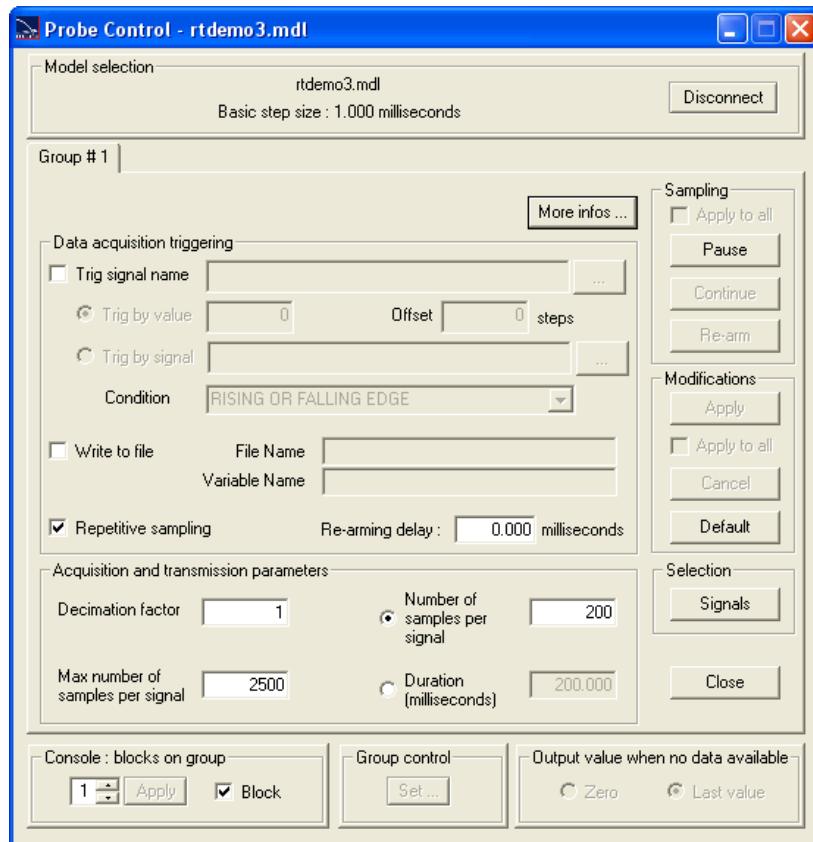


Figure 49:Probe Control

Following is a description of the parameters of the probe control.

Connect: Connects to a simulation model which acquisition parameters are to be changed. It displays the **Active Models** panel, if there are simulations running, to enable you to select the model from a list of running models.

Disconnect: Disconnects from the model we are currently connected to.

Group #: These tabs enable you to select which acquisition group parameters are to be changed.

More information: Opens the **group infos** panel on which additional information about the acquisition group is displayed.

Data acquisition triggering: Sets dynamically a trigger on the acquisition group. Data will be acquired only when the specified signal is triggered. The trigger is defined using the fields **Trig signal name**, **Trig by value**, **Offset**, **Trig by signal** and **Condition**. If a block OpTrigger is present in the model when activating the dynamic acquisition, the triggering will be set to the value of parameters provided via the Probe Control. When deactivating the dynamic triggering, the block OpTrigger will regain the triggering control once again.

Trig signal name: Sets the name of the signal to be triggered.

Trig by value: Specifies the value that must be reached by the trigger signal to trigger acquisition.

Trig by signal: Specifies a triggering signal instead of a constant value.

Offset: If this parameter is a positive value, it is used to set the number of steps (cycles) to wait before starting the acquisition system, once the trigger condition is met. The default value is 0, which means start as soon as the trigger condition is met. The trigger offset can also be a negative value n. In this case, the n samples which correspond to the simulation results of the n steps before the trigger condition happened will be acquired (up to the number of samples per signals). The range value for the offset is [-FrameSize ; +FrameSize]. See **Number of samples per signal** and **Duration** definitions hereafter for the definition of the frame length.

Condition: Condition to be satisfied so that the acquisition system is triggered. The different types of condition are:

- Rising or falling edge,
- Rising edge (the threshold is crossed on a positive slope, that is from below),
- Falling edge (the threshold is crossed on a negative slope, that is from above),
- Trigger signal \geq trigger level,
- Trigger signal \leq trigger level.

Write-to-file: Saves acquisition group signals on the target node's hard drive rather than sending them to the Windows NT Console on the host computer. Signals are saved in MATFILE version 4, using the name supplied in the **FileName** field.

Before **write-to-file** is enabled for the first time, the **FileName** and **Variable Name** fields are empty.

The **FileName** and **Variable Name** fields may contain valid names even if the group is not set to **write-to-file**. This is because this panel uses the filename set for the group from the **ProbeControl** panel. Disabling the **write-to-file** option from the **ProbeControl** panel does not clear these fields.

As more than one subsystem may be running on the same target node, RT-LAB adds the group number, the subsystem name, and the subsystem id to the filename specified by the user on the **Probe Control** panel. This is a simple way to avoid file name collisions on target.

FileName: Name of the file in which acquisition signals are to be saved when the **Write-to-file** option is checked.

Variable Name: Name of the variable to be associated with the signals saved in **FileName**.

Repetitive sampling: Enables you to choose repetitive (automated) sampling or non-repetitive (manual control) sampling. If you choose non-repetitive, the **Re-arm** button enables you to manually re-arm data acquisition.

Rearming delay: This is the acquisition delay between the end of one frame and the beginning of the next. This parameter is used only for repetitive sampling.

Decimation factor: Lets you skip acquisition during some calculation steps. When this parameter is set to a number **n**, a sample is acquired at every **n-th** calculation step.

Number of samples per signal: Defines the frame length of the signals for a group according to the number of samples for each signal. These signals are received and saved in a buffer, ready for transmission to the Console via the Ethernet link (or other communication devices). For example: if n is the number of samples per signal of a group with m signals, the buffer's size value of $m \times n$, and the signal frame have a length value of **n** corresponding to **n** cycles (calculation steps) up to the maximum number of samples.

In other words, this guarantees that there is no data loss for **n** simulation steps. The default value = max (1, 200 / model / timestep / decimation / 1000).

Max number of samples per signal: Defines the maximum number of samples that can be acquired for a specified acquisition group of the current model. Setting this value prior to simulation ensures that no dynamic memory allocation (a process that can degrade real-time performance and stability) takes place during simulation. This maximum is the total number for all the signals in the group. For example, when acquiring 4 signals for an acquisition group, you can specify a number of samples per signal up to the number set here.

Duration: Defines the length of time (in milliseconds) during which acquisition of signals is performed; this parameter provides another possibility for defining the frame length, and is linked to the decimation factor and the number of samples per signal using this formula:

$$\text{Duration} = (\text{Decimation factor}) \times (\text{Number of samples per signal}) \times (\text{Basic calculation step}).$$

Sampling, Pause: Enables you to interrupt a signal acquisition and transmission without affecting the model, which continues to run. This parameter is used with repetitive sampling.

Sampling, Continue: Enables you to resume data acquisition after a pause. This parameter is used with repetitive sampling.

Sampling, Rerarm: Enables you to manually rearm the acquisition system. This parameter is used with non-repetitive sampling.

Sampling, Apply to all: Applies the current sampling options to all acquisition groups.

Console blocks on group: This multiple-choice function (Groups 1 to n, or without blocking) enables you to stop signal display devices from the scope blocks (Simulink scopes, LabVIEW panels, etc.) when data acquisition is not available. But because all Simulink windows are frozen in blocking mode, a **Do not block** option is added to avoid completely freezing Simulink. In this mode, and when acquisition is stopped, the value displayed is the one chosen by the **Output** value when no data available parameter.

Output value when no data available: When the **Do not block** option is selected, this parameter enables you to choose an output value in the absence of data (zero or the last output value).

Set group control: Enables multiple users to connect to the same model and control acquisition parameters. This function enables each user to select just the acquisition group(s) for which they would like to control parameters; users who subsequently connect to the same model can select among the remaining acquisition groups.

Modifications, Apply: Used to validate modifications to panel parameters.

Modifications, Apply to all: Applies modifications to all acquisition groups.

Modifications, Cancel: Enables you to cancel the modifications to panel parameters.

Default: Sets all acquisition control parameters to their default values.

Note: Please note that the **Number of Samples per Signal** and **Duration** are set by default to values that allow simulations at 200 ms or 5Hz.

More information: Opens the **group infos** panel where you can find additional information on about the acquisition group.

Signals: Opens the dynamic signals panel to enable you to add signals to an acquisition group.

Close: Closes the **ProbeControl** panel. If one of the parameters on the panel is modified without being validated with the **Apply** button, closing the panel does not validate the modification made to that parameter.

Group Control panel

For a model with more than one acquisition group, the **GroupControl Panel** is displayed when the **ProbeControl** panel is opened.

To open the probeControl panel, select the model and assure you that it is built. Then, open the panel from the **Tools > Probe Control** menu. The probeControl will be automatically connected to this model. In all other cases, you are required to click on **Connect** to choose a model from a list of running simulations. The **ProbePanel** is then said to be standalone. In this configuration, it receives some important notifications from the target nodes such as System Paused, System Reset, System Error, etc.



Figure 50:GroupControl Settings

Group Information panel

The **Group info** panel displays additional information about the configuration and settings of an acquisition group.

Subsystem Name	Logical Id	Nb. Sign...	Target Node	File Name	Variable Name
sm_controller	1	1	localhost	MyFile_Gr1_Gr1_controller_1.mat	MyVar_Gr1_Gr...
ss_plant	3	1	localhost	MyFile_Gr1_Gr1_plant_3.mat	MyVar_Gr1_Gr...

Figure 51:Group info

Following is a description of the parameters of the dialog:

Subsystem Name: Name of target subsystems.

Logical Id: Logical Id of corresponding subsystem: assigned by RT-LAB.

Nb Signals: Number of signals that the corresponding subsystem sends to the group.

Target Node: Name of the target node to which the corresponding subsystem is assigned.

File Name: Name of the file saved by the corresponding subsystem on the target when writing to file.

Variable name: Name of the variable associated with the file if **write-to-file** is enabled.

Dynamic Signals panel

The Dynamic Signals panel is the graphical interface to manage dynamic signals. The dynamic signals are the signals available inside the model. A signal is the port's outputs of a block. Only double-type signals could be added to an acquisition group in order to be displayed.

In order to have access to all the signals inside them the model, the option "Reuse storage signals" should be set to off in the Simulation parameters of the Simulink model.

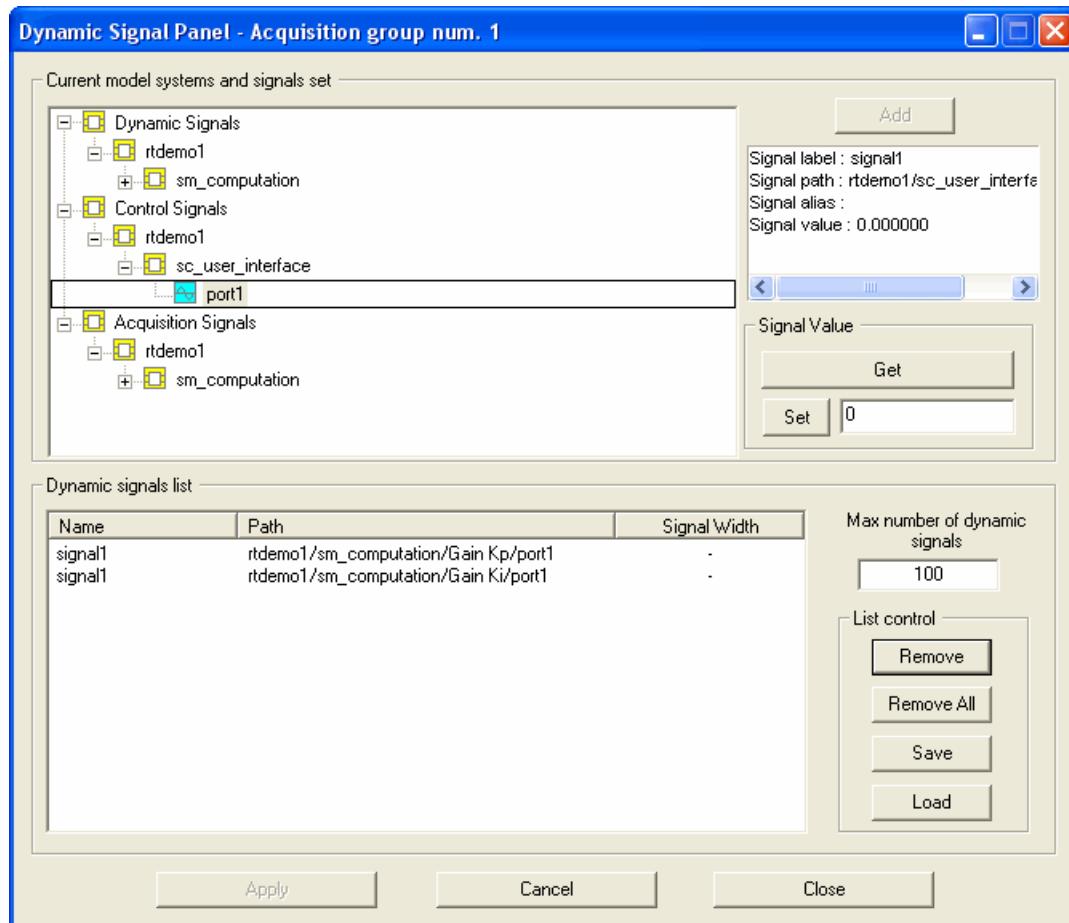


Figure 52:Dynamic Signal panel

Following is a description of the parameters of the dialog:

Current model systems and signal set: Signals tree containing three groups of signals:

- Dynamic signals: any signals inside the model.
- Control signals: signals that are sent from the console (SC_ subsystem) to the computation subsystem (SM_ or SS_).
- Acquisition signals: signals sent from the computation subsystem to the console.

Dynamic signals list: List of dynamic signals added to the current acquisition group.

Add: Adds the selected dynamic signal in the tree to the current acquisition group.

Signal value - Get: Gets the value of the selected signal in the tree.

Signal value - Set: Sets the value of a control signal.

Max number of dynamic signals: Maximum number of signals that could be added to the acquisition group while the simulation is running.

Remove: Removes a signal from the dynamic signals list.

Remove All: Removes all the signals from the dynamic signals list.

Save: Saves the current list.

Load: Loads a saved list.

Signal Selection panel

The Signal Selection panel is the graphical interface to select signals while setting dynamic triggering. The dynamic signals are the signals available inside the model. A signal is the port's outputs of a block.

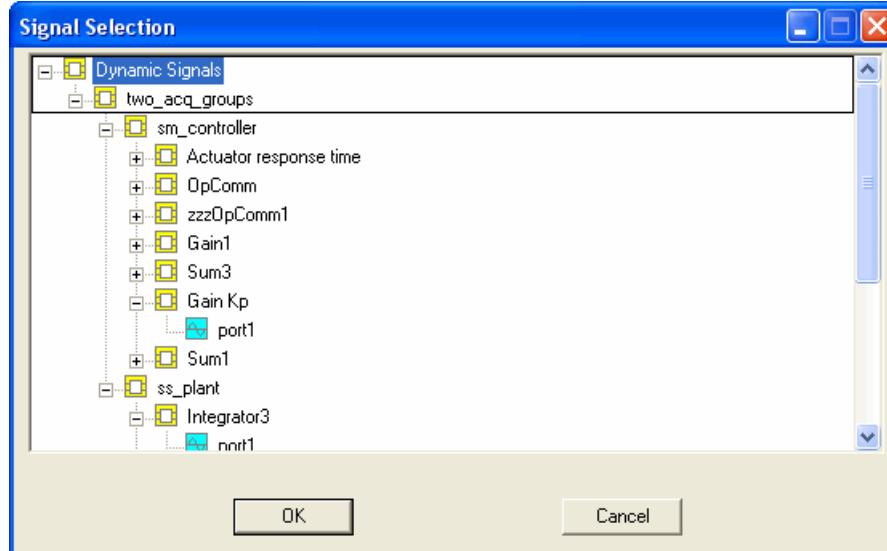


Figure 53:Signal Selection panel

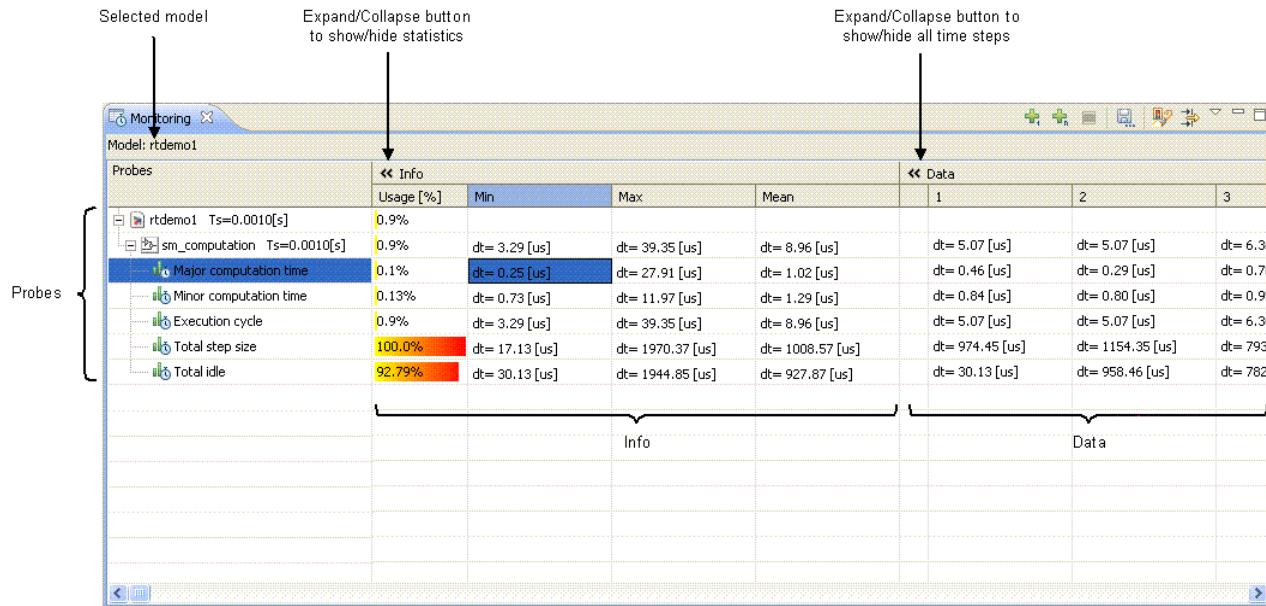
Monitoring View

The Monitoring view displays CPU usage for each task/probe executed during real-time simulation. It displays the duration of each probe and can display its start and stop time. It detects overruns and highlights steps where overruns are detected. It enables you to save the results and to export them to other software, such as Excel.

Monitoring your model helps you to profile your CPU usage and your tasks and to distribute your simulation by evaluating time spent by each part of the simulation. See [Monitoring Models](#) for more information about monitoring.

To show the monitoring view in the current perspective, click **Window > Show view > Other...** menu and the select the **RT-LAB > Monitoring** item.

Here is what the Monitoring view looks like. Details about each area are explained below.



Probes area

This area shows all probes currently visible. It also shows the model and subsystems probes that summarize the tasks for the model and the subsystems. By default, only commonly used probes are visible. To select which probes are visible, use the Filter dialog to edit and select filters.

Info area

This area shows the main statistics of all visible probes. By default, only the CPU usage [%] is visible but it is also possible to show the minimum, maximum and mean durations of all visible probes for the current acquisition frame using the expand/collapse buttons of this area.

Data area

This area shows the duration, start time and stop time for all visible probes for each time step of the acquisition frame. The expand/collapse buttons of this area show/hide the time steps with/without overruns. By default, only the steps with overruns are visible, and only the durations are visible in the cells but it is possible to add timing content from the view menu. When a time step contains an overrun, a red symbol appears at the top of the column.

Using the Monitoring View

Here are the steps to use the Monitoring view. See the following sections for more detail on the toolbar and menu items.

- Select the model you want to monitor in the **Project Explorer** view.
- Open the Monitoring view if necessary. Click **Window > Show view > Other...** menu and then select the **RT-LAB > Monitoring** item.
- While the model is running, click the **Get Probes** or **Get Probes constantly** buttons to perform 1 or N monitoring acquisitions respectively. If consecutive acquisitions are performed, click the Stop button to cancel the acquisition.
- In the probe area, expand and collapse model and subsystem probe items to display the probes of interest. If necessary, click the **Filter** button in the toolbar to apply different filters and change which probes are visible. By default, only the most commonly used probes are displayed.
- Expand and collapse the Info and Data area to get more details on the probes.
- Click the **Units** menu item and then use the Units dialog to change the units of all timing information.
- Click the **Properties** menu item and then use the Monitoring Properties dialog to change the acquisition setting of the monitoring.
- Search for time step with overrun in the data area. To do so, use
 - the **Search overrun** button from the toolbar,
 - scroll through the column in the data area to find the error symbol or
 - collapse the column without overruns using the expand/collapse button from the data area.
- Save the monitoring results to a CSV file or copy the cells to your favourite text editor.

Toolbar

The view toolbar contains the following buttons:

ICON	DESCRIPTION
	Get probes Get probes from the running simulation. It acquires only one acquisition frame of N consecutive steps.
	Get probes constantly Get probes from the running simulation. It acquires consecutive acquisition frames until an overrun is detected or until the stop button is pressed.
	Stop probes Stop consecutive monitoring acquisition.
	Save as ... Save monitoring information as a CSV file. CSV file can be easily imported into an Excel worksheet.
	Show next overrun Find in the current acquisition frame the next step where an overrun occurred.
	Select probes Select the probes to be display in the monitoring view by selecting filters based on their name

Menu

The menu of the Monitoring view contains the following items:

DESCRIPTION
Select content > Start and Stop times
Select the content to be display in cells of all steps acquired during a frame. When the start and stop times are selected, the cell contains, in addition to the probe's duration, the probe's start and stop time.
Sort By
Allow sorting the probes in the monitoring view by name, by chronological order or by the default order
Units
Open a dialog that allows configuring the units used in the monitoring view.
Properties
Open a dialog that allows configuring the main acquisition setting of the monitoring view.

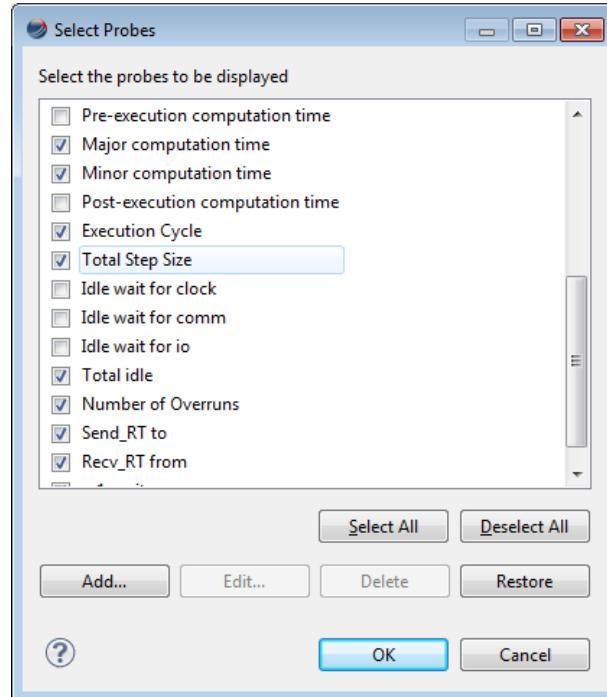
Context Menu

The context menu of the Monitoring view contains the following items:

DESCRIPTION
Copy
Copy the selected cell to clipboard
Delete
Delete the selected probe from the list of visible probes. When delete a filter is automatically created.

Filter Dialog

This dialog allows you to select the probes that will be displayed in the Monitoring view.



Select All

This button allows you to select all filters from the list.

Deselect All

This button allows you to deselect all filters from the list.

Add...

This button allows you to create new filter. It is useful when custom probe are inserted in the model.

Edit...

This button allows you to edit a filter and then change its name or its matching regular expression. For advanced users only.

Delete...

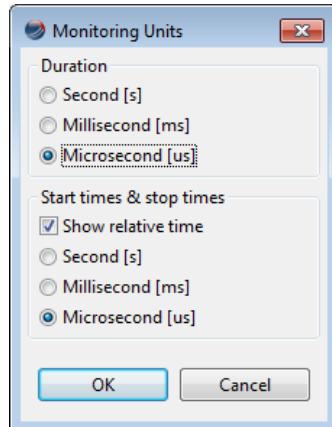
This button allows you to delete a filter from the list.

Restore

This button allows you to restore all filters to their default values.

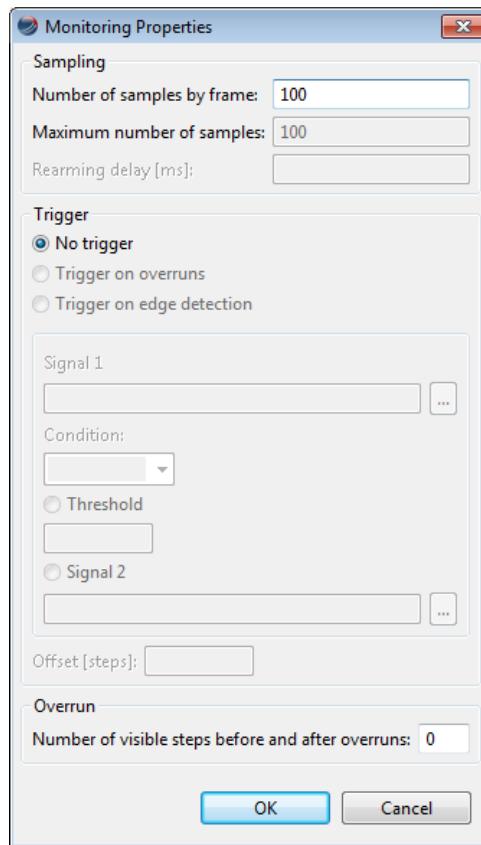
Units Dialog

This dialog allows configuring of the units of the durations and start and stop times shown in the Monitoring view. Available units are seconds, milliseconds and microseconds. It is also possible to configure whether the start and stop times shown are relative to the beginning of the current acquisition frame or based on the absolute time of the simulation.



Monitoring Properties Dialog

This dialog allows you to change the acquisition settings for the monitoring.



Number of samples by frame

This setting allows you to change the number of consecutive steps (samples) by frame that are acquired by the monitoring. This setting can be changed while the simulation is running.

Maximum number of samples

This setting allows you to determine the maximum number of samples that could be set for one frame. This setting is enabled when the model is not running and was properly compiled.

Trigger

The trigger settings are not yet available from this dialog.

Number of visible steps before and after overruns

It determines the number of steps that are visible before and after a time step with overruns. This setting is used when steps without overruns are hidden from the data area (using the expand/collapse button if the data area). It allows to focus on steps close to overruns.

Related Concepts

[Monitoring Models](#)

Terminal View

The **Terminal View** provides a quick access to the remote target by opening a telnet connection. From here, it is possible to gather information by sending commands to the target. This view is very useful for diagnostic and debug tasks.

The **Terminal View** appears in a tabbed notebook:

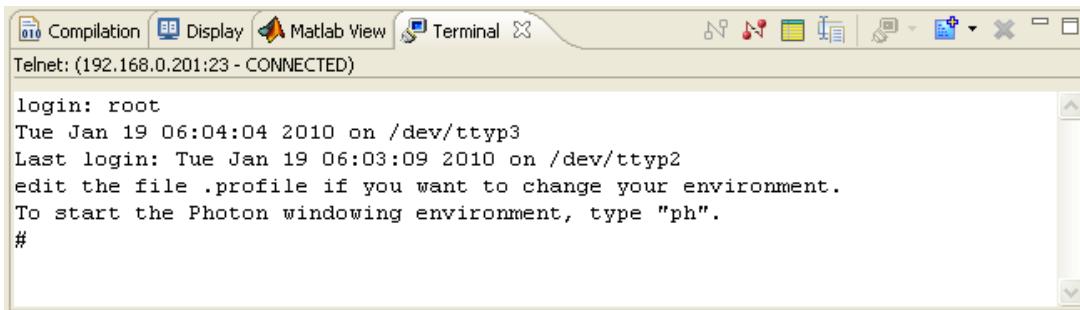


Figure 54:Terminal View

The next section presents how to connect to remote target. Other sections show some examples and explain the toolbar and the contextual menu.

By default, the **Terminal View** is not included in the Edition perspective. To add it to the current perspective, click Window > Show view > Other... > Terminal > Terminal

How to connect to remote target

Open the Terminal View

To open a new **Terminal View** on a specific target, click on the Telnet menu item from the Tools menu of the target.

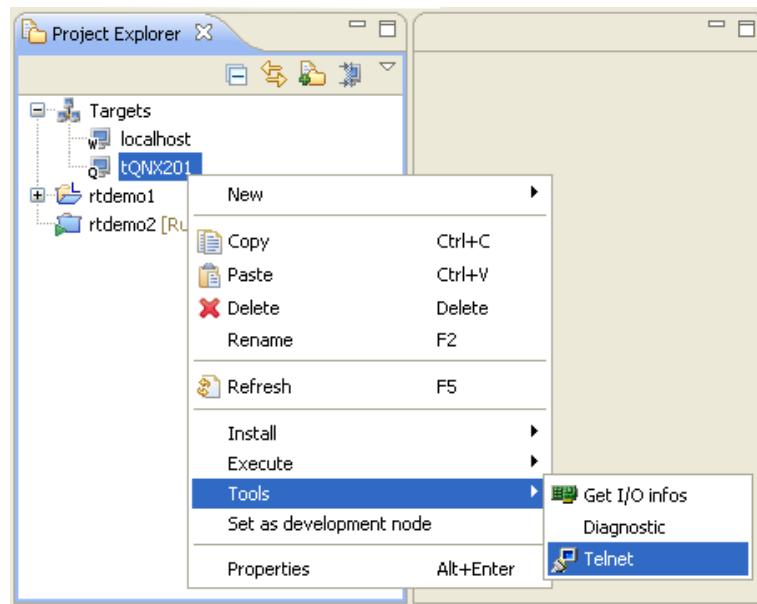


Figure 55:Open Terminal view

Verify the terminal settings and then click OK. The view title settings can be changed if more than one terminal have to be opened at the same time.



Figure 56:Terminal Settings

Then, verify the connection status at the top left of the terminal view. It should be updated to CONNECTED. Ex.: Telnet: (192.168.0.201:23 - CONNECTED)

Login to remote target

To login to a remote target, you need to enter the login name and the password.

By default, the user name for a QNX target is "root" and there is no password. The user name for a Red Hat target is "root" and the password is "redhat". The user name for a RedHawk target is "root" and the password is "redhawk".

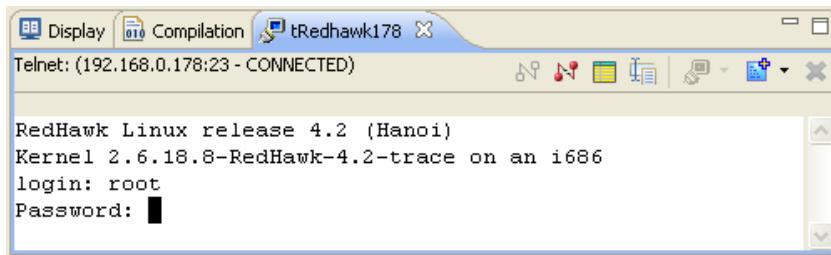


Figure 57:Login with Terminal view

What does the Terminal View offer

The **Terminal View** gives the opportunity to communicate with targets to:

- Watch the running processes
- Know which versions of softwares are installed
- Launch configuration scripts
- Install new softwares

-
- Debug models
 - Consult information logs
 - Open crash files
 - Execute operating system commands

Note: All Red Hat commands are available on RedHawk target.

List of commands for QNX and Red Hat targets

Command	Description
/usr/opalrt/common/bin/flash_update -bim	This is a request to get Board Identification Messages (BIM) of remote SignalWire boards. It searches and displays data on installed board.
ls -lo /usr/opalrt	Lists the RT-LAB versions installed on the target.
uname -a	Prints the name, version and other details about the current target and the operating system running on it.
ps -A	Reports the process status.
ifconfig -a	Allows the user to view information about the configured network interfaces.
ls /dev	Lists the devices that are currently running on the target.

List of commands specific to QNX targets

Command	Description
pidin info	Displays information about the processors installed on the target.
pci -v	Prints information on PCI card installed on the target. Use the command pci -v grep 'Device ID' to get list of the device IDs.
ls -lo /var/dumps	Lists the dump files that are written by the dumper utility. Whenever a program terminates abnormally, a dump of the current state of the program is written to disk. The dump filename is the same as the program name with a .core extension.

List of commands specific to Red Hat targets

Command	Description
cat /proc/cpuinfo	Displays information about the processors installed on the target.
lspci -v	Displays information about all PCI buses in the system and all devices connected to them.

List of commands specific to RedHawk targets

Command	Description
cpu	Displays tabular information about the processors installed on the target.

List of commands specific to Windows targets

Command	Description
dir C:\Opal-RT	Lists the Opal-RT directory
systeminfo	Displays information about the system
tasklist	Prints the name, the PID and the memory usage of all running applications and services on the target.
ipconfig /all	Allows the user to view information about the configured network interfaces.

Toolbar

The toolbar in the **Terminal View** contains the following buttons:

ICON	DESCRIPTION
	Connect Opens the connection to the target.
	Disconnect Closes the connection to the target.
	Settings Opens the Terminal Settings dialog. While connected, only the view title can be changed.
	Toggle Command Input Field Toggles the Command Input field to edit complex command lines on dumb terminals. See figure: Command Input Field.
	Display Selected Connections Selects a Terminal connection to show in this view instance. Only available when multiple connections have been defined in the view.
	New Terminal Connection in Current View Creates a new terminal connection in the current view.
	New Terminal View Creates a new terminal view that will be stacked in the tabbed notebook at the rightmost position.
	Remove Terminal Removes the currently selected Terminal Connection from the view. Only available when multiple connections have been defined in the view.

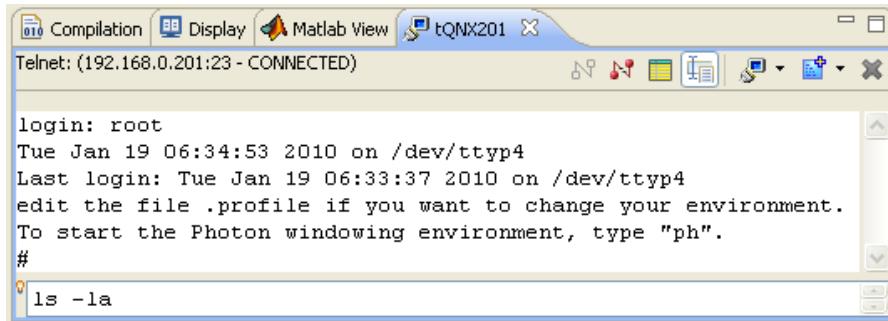


Figure 58:Command Input Field

Contextual Menu

To open the contextual menu, right-click on the output of the **Terminal View**.

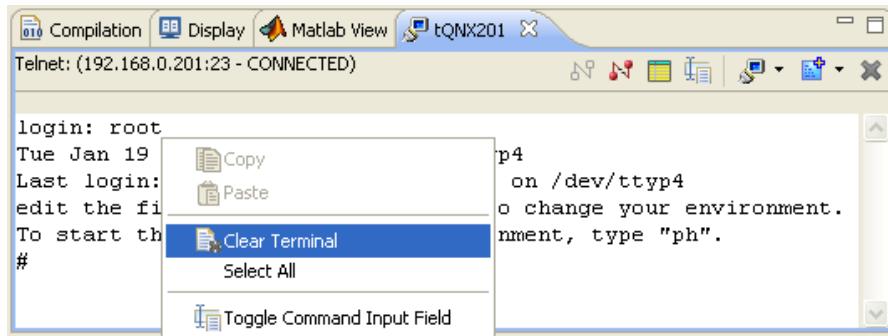


Figure 59:Contextual Menu

MENU ITEM	DESCRIPTION
Copy	Allows to copy selected text from the view.
Paste	Allows to paste text into the view.
Clear Terminal	Clears the text in the terminal view.
Select All	Selects all the text in the terminal view.
Toggle Command Input Field	Enables/disables the command input field at the bottom of the Terminal view.

Related concepts

- [Views](#)
- [Perspectives](#)
- [Toolbars](#)

Related tasks

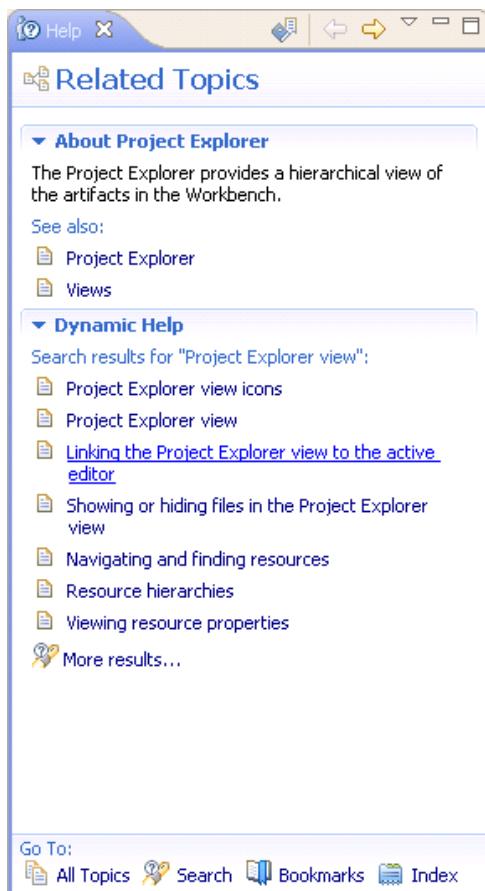
- [Debugging](#)
- [Opening views](#)
- [Moving and docking views](#)

Related reference

Help view

The Help view provides user assistance inside the workbench. The view consists of several pages. Each page presents help in a different fashion. Hyper links at the bottom of the help view allow switching among pages, and clicking any topic will display its contents.

Related Topics



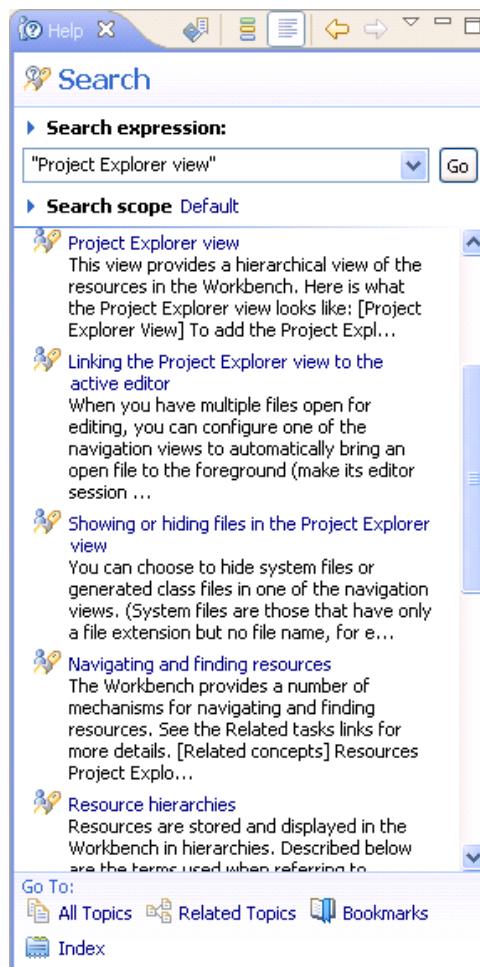
The Related Topics page shows description and help topics related to the current workbench context. The About section shows context help specific to your current context, and the Dynamic Help section shows some search results that may be related. For example, if the currently active workbench part is the Project Explorer view in the RT-LAB Edition perspective, local help will find and present topics describing "Project Explorer view" or "Edition perspective". The Related Topics page tracks changes in the workbench and continuously updates displayed information.

All Topics



The All Topics page shows the table of contents. It is a hierarchy of all help topics arranged in a tree. The tree branches can be expanded to browse topics that can be displayed using a single mouse click.

Search



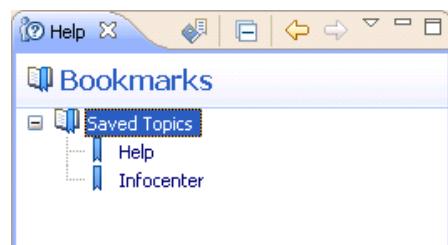
The Search page allows locating local topics, cheat sheets, welcome content, remote documents, and other documents given a search query. Links to search hits are displayed along with a summary of topic contents. Search scope controls a subset of documentation being searched. Multiple search scopes can be configured, each defining a custom set of resources from among local documentation, additional local search engines or remote engines on the web.

Index

The Index page provides an index of keywords that direct the user to specific help topics, similar to indexes found at the back of a book. As you type in the text field, the best match will automatically be highlighted in the list of keywords. Pressing enter or clicking on a keyword will display the given topic.

Note: The index page will only appear when index content is available in the workbench.

Bookmarks



The Bookmarks shows topics marked as personal bookmarks.

Toolbar

The toolbar of the Help view contains the following buttons. The available buttons depends on the currently displayed page.

ICON	DESCRIPTION
Back	Displays the page or the topic that was displayed immediately prior to the current display.
Forward	Displays the page or the topic was displayed immediately after the current display.
Collapse All	Collapses the tree expansion state of all topics on the page.
Show All Topics	When filtering of workbench elements is enabled, the buttons enables displaying topics for disabled elements.
Show Result Categories	Switches between sorting search results by relevance and by logical containers for example books.
Show Result Descriptions	Displays short description of each search hit, when available.
Menu	Provides menu items that allow you switching help view pages.

Related concepts

[Help](#)

Search View

This view shows the results of a search. The same view is shared for different types of searches, like [File Search](#) and [RT-LAB Search](#).

RT-LAB Search results

RT-LAB searches will only search for expressions in RT-LAB objects specified in the **RT-LAB Search** dialog.

Here is what the results of [Search View](#) look like for an **RT-LAB Search**:

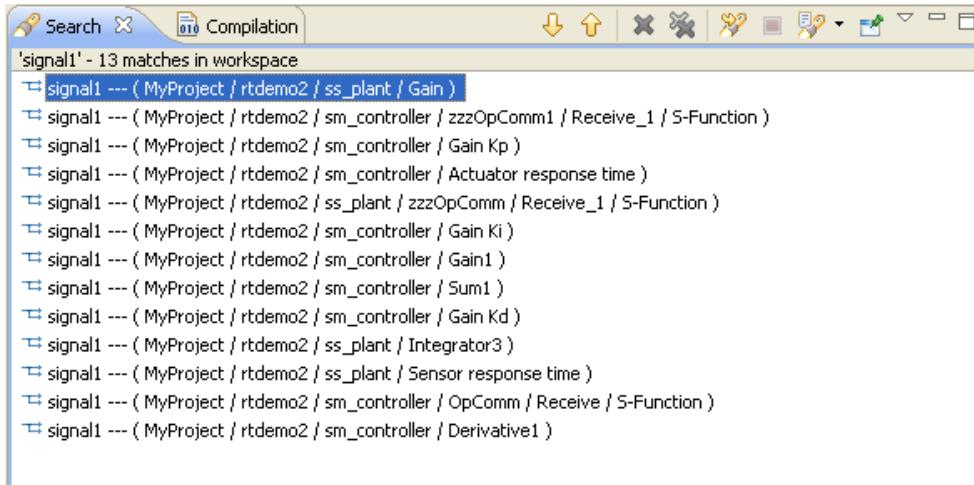


Figure 60:Search View show as a List

If you close the [Search View](#), you can return to it later by selecting Window > Show View > Other... > General > Search.

Display the results as a tree

To enable the display of the results as a tree, select the option **Show As Tree** from the view menu.

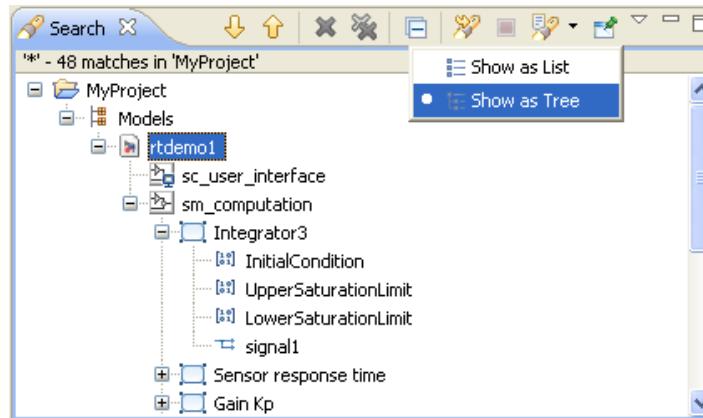


Figure 61:Show results as a Tree

Working with search result items

Drag and drop items:

- Drag and drop features in the editor are enabled for the model and the subsystem objects. If the search found a model or subsystem, select it, drag it to the editor and drop it. The model will open in the editor.

Double click on items:

- Double clicking on a model or a subsystem opens the editor.

View / Edit variable from the contextual menu:

- This option helps to see the value of the selected items by opening the Variable Viewer.

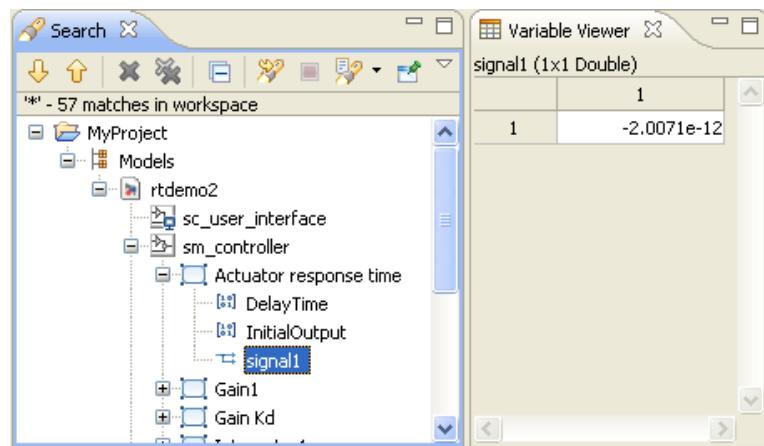


Figure 62:Variable Viewer

Assign Working Set from the contextual menu:

- This command is used to assign the selected object to an existing or a new Working Set. The next figure shows the Variables Table the plant_response signal that have been added to the Working Set 1.

The screenshot shows the 'Variables Table' window. The title bar indicates 'Working Set: WorkingSet1'. The table has columns: Name, Path, Value, Units, Min, and Max. There is one row for 'plant response' with the value '2.4491e-12'. At the bottom of the window, there is a status bar with the text 'Filter: [] Search: [] RE | 1 Loaded - 1 Shown - 0 Selected'.

Name	Path	Value	Units	Min	Max
plant response	rtdemo2/ss_plant/Integrator2/port1	2.4491e-12		-Infinity	Infinity

Figure 63:Variable Table

Contextual Menu for RT-LAB Search

To open the contextual menu, right-click on one of the matches.

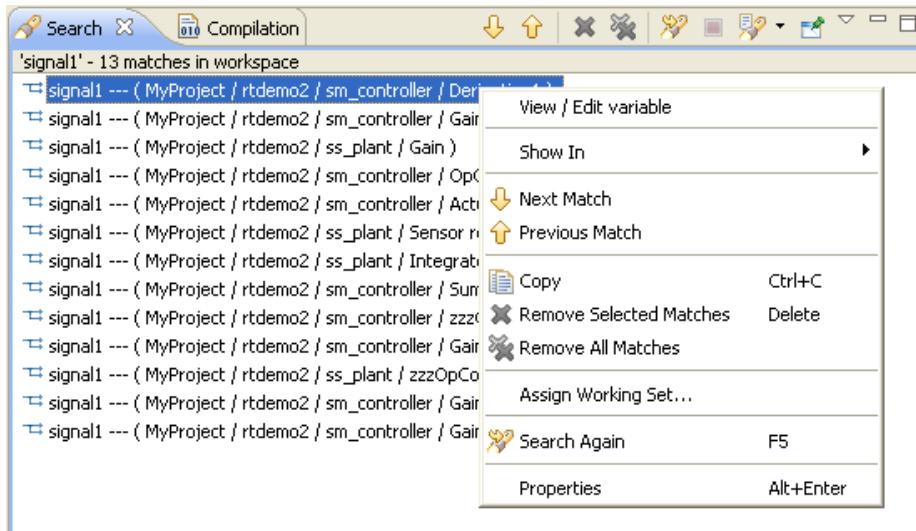


Figure 64:Contextual Menu

MENU ITEM	DESCRIPTION
View / Edit variable	This command is used to open the variable viewer for the selected object.
Show In	This command is used to show the selected object in a view, only if applicable.
Next Match	This command highlights the next match of the search expression in the search view.
Previous Match	This command highlights the previous match of the search expression in the search view.
Copy	Copies the text of the selected match to the clipboard.
Remove Selected Matches	Removes all highlighted matches from the search results.
Remove All Matches	Removes all search results from the search view.
Assign Working Set...	This command is used to assign the selected object to an existing or a new Working Set.
Search Again	This command reruns the current search again, so that removed search results reappear or changes are reflected.
Properties	Opens the Property View for the selected object.

File Search results

Text searches will only search for expressions in file specified in the **File Search** dialog.

This example shows results for the search "it". The title of the Search view shows that four matches were found.

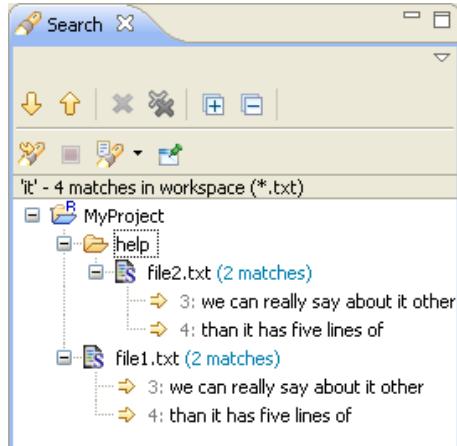


Figure 65:File Search result

Within the Search view two files are shown and within each file there were 4 matches found.

- Click the **Show Next Match** button to navigate to the first match of the search expression ("it"). Notice that the file file1.txt is automatically selected and opened in the editor area. Click **Show Next Match button** two more times. Once again the Search view automatically opens the file (file2.txt).
- It is sometimes useful to remove uninteresting matches from the search results. The Search view's popup menu allows you to do this using **Remove Selected Matches** which removes any selected file entries (and all matches in them) from the Search view. Note that this **only** removes the entries in the Search view, it does not affect the files themselves. Select file1.txt and choose **Remove Selected Matches** from the popup menu. The Search view now shows only the matches for file2.txt
- Perform a second search for "that" by clicking on the Search button in the Workbench's toolbar.
- The Search view updates to show the results of the new search. Use the drop down button on the Search view's toolbar to move back and forth between the two search results.
- In the drop down button choose **'it' - 2 match in workspace**. The Search view switches back to show the original search. On the context menu choose **Search Again** to repeat the initial search. Notice that once again there are four matches.

So far, you have seen how to manage your search results and how to switch between different searches. However, it might happen that you do not want the search view to change even if further searches are performed. For this you can **pin** the search view, which causes subsequent searches to be shown in a second Search view.

Toolbar

The toolbar in the **Search View** contains the following buttons:

ICON	DESCRIPTION
	Show Next Match This command highlights the next match of the search expression in the search view.
	Show Previous Match This command highlights the previous match of the search expression in the search view.
	Remove Selected Matches Removes all highlighted matches from the search results.
	Remove All Matches Removes all search results from the search view.
	Collapse all Collapses every tree item in the search view.
	Run the Current Search Again This command runs the current search again, so that removed search results reappear or changes are reflected.
	Cancel Current Search Cancels the current search.
	Show Previous Searches This command allows you to browse previously conducted searches and repeat a previous search. You can select a previous search from the drop-down menu or clear the search history.
	Pin the Search view Pinning the search view means that subsequent searches will show their results in another search view and that the pinned view remains unchanged.

Related concepts

[Views](#)
[Perspectives](#)
[Toolbars](#)

Related tasks

[Debugging](#)
[Opening views](#)
[Moving and docking views](#)

Related reference

[RT-LAB Search](#)
[File Search](#)

Bookmarks view

The Bookmarks view displays user defined bookmarks.

To add the Bookmarks view to the current perspective, click **Window > Show View > Other... > General > Bookmarks**.

The Description column contains a description of the bookmark. You can edit the description by selecting **Properties** from the context menu.

The Resource and Path columns provide the name and location of the resource associated with each bookmark.

The Location column indicates the line number of the bookmark within its associated resource.

Toolbar

The toolbar of the Bookmarks view includes the following buttons.

Delete

Delete the selected bookmark.

Go to

Open the bookmark's resource and navigate to the bookmarked region.

Menus

Click the icon at the left end of the view's title bar to open a menu of items generic to all views. Click the black upside-down triangle icon to open a menu of items specific to the Bookmarks view. Right-click inside the view to open a context menu.

Creating a bookmark within a file

The Workbench allows you to create bookmarks in files that you edit so that you can quickly reopen those files from the Bookmarks view.

With the file open in an editor, right-click in the gray border at the left of the editor area, next to the line of code or text that you want to bookmark.

Select **Add Bookmark** from the pop-up menu.

Notice that an icon for the bookmark now appears in the left border of the editor area. A line is also added to the Bookmarks view.

You can reopen the file for editing at any time by double-clicking the bookmark in the Bookmarks view.



Tasks view

The Tasks view displays tasks that you add manually. You can associate a task with a resource in the Workbench, but this is not required.

By default, the Tasks view is not included in the RT-LAB Edition perspective. To add it to the current perspective, click **Window > Show View > Other... > General > Tasks**.

The first column indicates whether the task is completed. Completed tasks are flagged with a check mark, which you add manually.

The second column indicates whether the task is high, normal, or low priority.

The Description column contains a description of the line item. You can edit the description of user-defined tasks by selecting Properties from the context menu.

The Resource and Path columns provide the name and location of the resource associated with each line item.

The Location column indicates the line number of the line item within its associated resource.

Toolbar

The toolbar of the Tasks view includes the following buttons.

Add task

Manually add a "to do" item to the Tasks view.

Delete

Delete the selected line item.

Filter

Filter the view according to the type of item.

Menus

Click the icon at the left end of the view's title bar to open a menu of items generic to all views. Click the upside-down triangle icon to open a menu of items specific to the Tasks view. Right-click inside the view to open a context menu.

Adding line items in the Tasks view

The Tasks view contains line items for system-generated problems, warnings, and errors. You can add your own entries to the table to build a list of to-do items, or tasks.

- On the toolbar in the Tasks view, click the New Task button . The New Task dialog will open.
- Type a brief description for the task in the Description field. The new task is assigned default priority and completed values. These values may also be modified within the New Task dialog.
- Press OK.

Associating a task with a resource

You can associate tasks with an editable resource, for instance to remind yourself to update a line of source code later.

-
- In the project explorer view, double-click the resource with which you wish to associate the new task. The resource opens in the editor area.
 - Right-click in the gray border at the left of the editor area, beside the line of text or source code against which you want to log the new task.
 - On the pop-up menu, select **Add Task**.
 - When prompted, enter a brief description of the task.

A new task icon appears in the border of the editor area, to the left of the line where you added the task. When you move the mouse pointer over the marker, the description of the task is displayed as a tooltip. The task is also added to the Tasks view. You can delete a task either by right-clicking its icon in the editor area and selecting **Remove Task**, or by pressing the Delete key in the Tasks view.

Internal Web Browser view

This view allows you to display web pages. By default, it is used to display web pages when they are opened from the workbench. You can modify this setting in the [**Web Browser preference page**](#).



Markers view

This view shows problems, tasks and bookmarks that have been added to your resources.

By default, the Markers view is not included in the RT-LAB Edition perspective. To add it to the current perspective, click **Window > Show View > Other... > General > Tasks**.

The Description column contains a description of the line item. You can edit the description of user-defined tasks by selecting Properties from the context menu.

The Resource and Path columns provide the name and location of the resource associated with each line item.

The Location column indicates the line number of the line item within its associated resource.

The Type column indicates the type of marker of the line item: problem, task, bookmark.



Navigator view

This view shows resources in the workspace. It displays projects and other resources that you are working with.

Related concepts

[Project Explorer](#)



Outline view

The Outline view displays an outline of a structured file that is currently open in the editor area, and lists structural elements. The contents of the Outline view are editor specific. The contents of the toolbar are also editor specific.

To add the Outline view to the current perspective, click **Window > Show View > Other... > General > Outline**.



Problems view

As you work with resources in the workbench, various tools may automatically log problems, errors, or warnings in the Problems view. When you double-click the icon for a problem, error, or warning, the editor for the associated resource automatically opens to the relevant line of code.

By default the problems view will group your problems by severity. You can also group them by type or not at all. Certain components will add their own grouping. The grouping can be selected using the Group By menu.

The first column of the Problems view displays an icon that denotes the type of line item, the category and the description. Left-click the item to open the file in an editor and highlight the line containing the problem.

You can configure the contents of the Problems view to view only warnings and errors associated with a particular resource or group of resources. This is done using the Configure Contents dialog available from the drop down menu. You can add multiple filters to the problems view and enable or disable them as required. Filters can either be additive (any problem that satisfies at least one of the enables filters will be shown) or exclusive (only problems that satisfy all of the filters will be shown). The two most popular filters (All Errors and Warnings on Selection) are provided by default.

Problems can be fixed by selecting Quick Fix from the context menu. The list of possible resolutions will be shown.

To add the Problems view to the current perspective, click **Window > Show View > Other... > General > Problems**.



Workbench Menus

The **Workbench Menus** contain menu items that provide access to all commands of RT-LAB.

Note that menu items could be added or removed by **Configuring perspectives** and menu items may appear or disappear when **Capabilities preference page** are enabled or disabled.

Menu items are grayed when the corresponding commands are disallowed. Commands are enabled according to:

- the active part of the RT-LAB Workbench, it means the Editor or the View that has the focus,
- the element that is selected in this active part,
- the state of this element (for example, a running Model cannot be loaded).

The RT-LAB workbench provides the following menus:

- **File menu**
- **Edit menu**
- **Navigate menu**
- **Search menu**
- **Simulation menu**
- **Tools menu**
- **Window menu**
- **Help menu**

File menu

The **File** menu enables you to create, save, close, print, import, and export Workbench resources and to exit the Workbench.

New (Shift+Alt+N)

Enables you to create new resources by opening the corresponding wizard. Note that before you can create a new file, you must create a project in which to store the file.

Open File

Enables you to open a file for editing - including files that do not reside in the Workspace.

Close (Ctrl+W)

Closes the active editor. You are prompted to save changes before the file closes.

Close All (Shift+Ctrl+W)

Closes all open editors. You are prompted to save changes before the files close.

Save (Ctrl+S)

Saves the contents of the active editor.

Save As

Enables you to save the contents of the active editor under another file name or location.

Save All (Shift+Ctrl+S)

Saves the contents of all open editors.

Revert

Replaces the contents of the active editor with the previously saved contents.

Move

Enables you to move the currently selected resources to a different project.

Rename (F2)

Enables you to change the name of the currently selected resource.

Refresh (F5)

Refreshes the resource with the contents in the file system.

Convert Line Delimiters To

Alters the line delimiters for the selected files. Changes are immediate and persist until you change the delimiter again - you do not need to save the file.

Print (Ctrl+P)

Prints the contents of the active editor.

Switch Workspace

Opens the **Workspace Launcher**, from which you can switch to a different workspace. This restarts the Workbench.

Import

Launches the **Import** wizard, which enables you to add resources to the Workbench.

Export

Launches the **Export** wizard, which enables you to export resources from the Workbench.

Properties (Alt+Enter)

Opens the **Properties** dialog for the currently selected resource.

Recent file list

Contains a list of the most recently accessed files in the Workbench. You can open any of these files from the **File** menu by simply clicking the file name. You can control the number of files in this list from the Editors preference page.

Exit

Closes and exits the Workbench.

Edit menu

This menu helps you manipulate resources in the editor area and in the [Project Explorer](#) view.

Undo

This command reverses your most recent editing action.

Redo

This command re-applies the editing action that has most recently been reversed by the Undo action.

Cut

This command removes the selection and places it on the clipboard.

Copy

This command places a copy of the selection on the clipboard.

Paste

This command places the text or object on the clipboard at the current cursor location in the currently active view or editor.

Delete

This command removes the current selection.

Select All

This command selects all text or objects in the currently active view or editor.

Find/Replace

This command allows you to search for an expression in the active editor, and optionally replace the expression with a new expression.

Find Next

This command allows you to search for the next occurrence of the current selection, or for the next occurrence of the most recent expression found using the Find/Replace action.

Find Previous

This command allows you to search for the previous occurrence of the current selection, or for the previous occurrence of the most recent expression found using the Find/Replace action.

Incremental Find Next

This command allows you to search for expressions in the active editor. As you type the search expression, it will incrementally jump to the next exact match in the active editor. While in this mode, the up and down cursor keys can be used to navigate between matches, and the search can be cancelled by pressing left or right cursor keys, the enter key, or the escape key.

Incremental Find Previous

This command allows you to search for expressions in the active editor. As you type the search expression, it will incrementally jump to the previous exact match in the active editor. While in this mode, the up and down cursor keys can be used to navigate between matches, and the search can be cancelled by pressing left or right cursor keys, the enter key, or the escape key.

Add Bookmark

This command adds a bookmark in the active file on the line where the cursor is currently displayed.

Add Task

This command adds a task in the active file on the line where the cursor is currently displayed.

Word Completion

This action will attempt to complete the word currently being entered in the active editor.

Set Encoding

This action launches a dialog that allows you to change the file encoding used to read and write the file in the active editor.

Search menu

This menu provides advanced search tools.

Search... (Ctrl+H)

This command will open the Search dialog.

File...

This command will open the File tab of the Search dialog.

Text

This command searches for the currently selected text and displays results on the **Search View**.

The search may be performed within:

- **Workspace**
- **Project**
- **File**
- **Working set...** (opens a dialog to select a Working set)

Navigate menu

This menu allows you to locate and navigate through resources and other artifacts displayed in the Workbench.

Last Edit Position

This command allows you to jump the last edit position.

Go to Line

This command allows you to jump to a specific line in the active editor.

Tools menu

This menu provides access to external tools that interact with an RT-LAB simulation.

Probe Control

Opens the [Probe Control Panel](#) application for the currently selected Model.

ScopeView

Launches the ScopeView application to acquire and display signals from real-time simulation. See [Using ScopeView](#) for more information.

Open Matlab [version]

Opens the current version of Matlab. The version is displayed in the menu and could be changed in the [Matlab preference page](#) Preference page.

Python

Launches Python scripts, opens an interactive console or use the Macro recorder.

- **Run:** Starts the currently selected Python script.
- **Run configurations:** Opens a dialog to set the settings used when running a Python script.
- **Open Console:** Opens the [Interactive Python Console](#).
- **Record :** Record the next actions performed on a project in a new Python macro.
- **Stop recording :** Stop the recording of the Python macro and save it to file.
- **Pause recording :** Pause the recording.
- **Resume recording :** Resume the recording.
- **Cancel recording :** Cancel the recording..

Simulation menu

This menu allows to control a simulation.

Build (Ctrl+Alt+C)

Builds the selected Subsystems, the selected Models or all the Models of the selected Project. See

Building models**Build configurations...**

Opens the **Build Configurations** dialog.

Assign... (Ctrl+Alt+A)

Opens the **Assignment Page** for the selected Models.

Load (Ctrl+Alt+L)

Loads the selected Models or all the Models of the selected Project. See **Executing models**.

Execute (Ctrl+Alt+S)

Starts the execution of the selected Models or all the Models of the selected Project. See **Executing models**.

Execute a single step: Simulate one calculation step at a time of the model. (only available when the model is paused).

Pause (Ctrl+Alt+P)

Pauses the execution of the selected Models or all the Models of the selected Project. See **Executing models**.

Reset (Ctrl+Alt+R)

Resets the selected Models or all the Models of the selected Project. See **Executing models**.

Take Snapshot (Ctrl+Alt+T)

Takes a snapshot of the selected Model using the last snapshot's name and description. See **Taking a Snapshot** for more help on snapshot. This Model must be in Paused or Running state before taking a snapshot. See **Executing models**.

Take Snapshot As...

Opens a dialog to enter a name and a description of a snapshot:



This snapshot will then be taken for the currently selected Model. See [Taking a Snapshot](#) for more help on snapshot. This Model must be in Paused or Running state before taking a snapshot. See [Executing models](#).

Restore Snapshot (Ctrl+Alt+Y)

Opens a dialog to choose a previously taken snapshot:

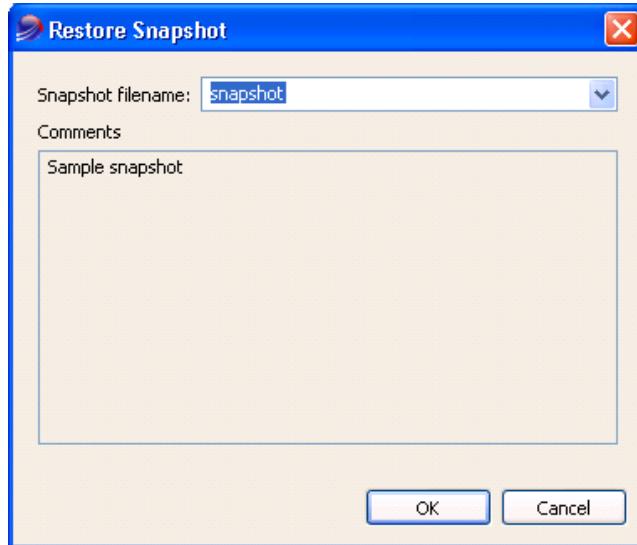


Figure 66:Restoring a Snapshot

This snapshot will then be restored for the currently selected Model. See [Taking a Snapshot](#) for more help on snapshot. The Model must be in the Paused state before restoring a snapshot. See [Executing models](#). Note that the snapshot may exist on the target directory where the model was loaded.

Load Parameters

Open the **Load Parameters wizard** that let you load a set of parameters and set values on the selected model.

Save Parameters

Open the **Save Parameters wizard** that let you save the values of all parameters of the selected model.

Window menu

This menu allows you to display, hide, and otherwise manipulate the various views, perspectives, and actions in the Workbench.

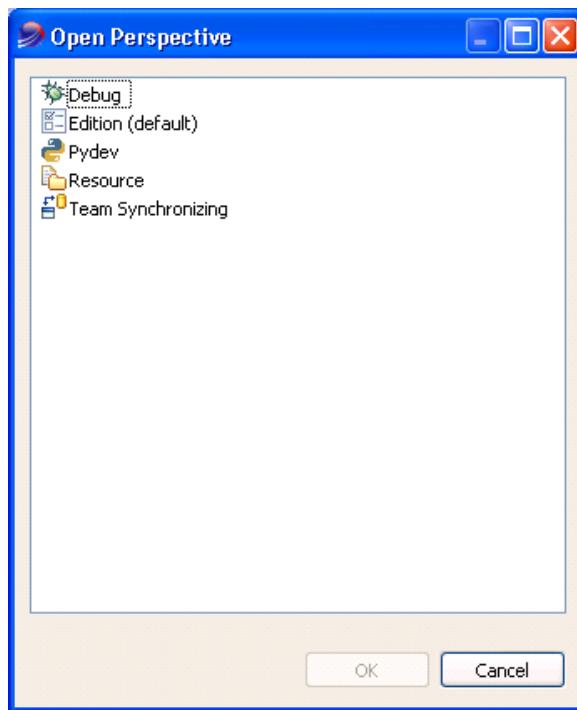
New Editor

This command opens an editor based on the currently active editor. It will have the same editor type and input as the original.

Open Perspective

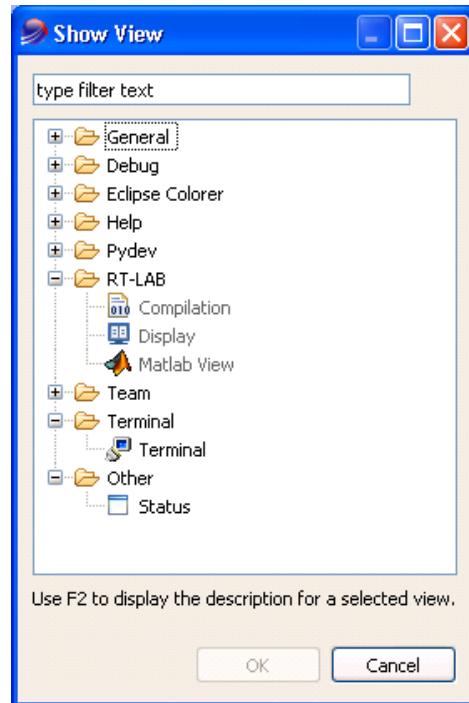
This command opens a new perspective in this Workbench window. This preference can be changed on the General > Perspectives preference page. All of the perspectives that are open within the Workbench window are shown on the shortcut bar.

The perspectives you will likely want to open are listed first. This list is dependent on the current perspective. From the Other... submenu you can open any perspective.



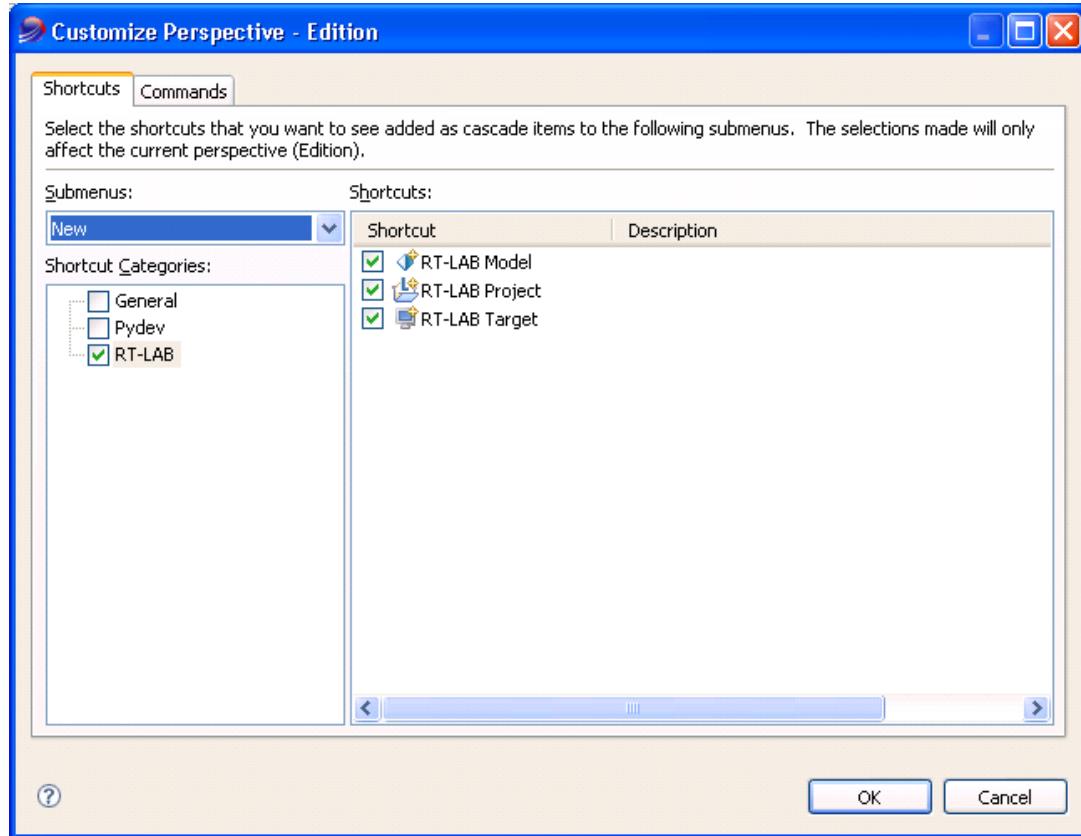
Show View

This command displays the selected view in the current perspective. You can configure how views are opened on the General > Perspectives preference page. Views you are likely to want to open are listed first. This list is dependent on the current perspective. From the Other... submenu you can open any view. The views are sorted into categories in the Show View dialog.



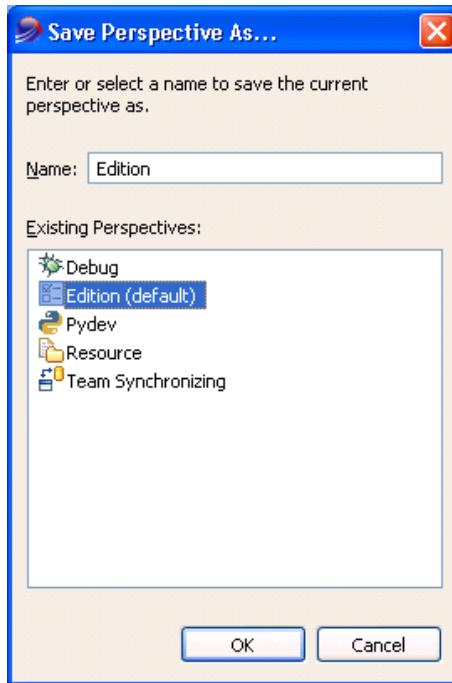
Customize Perspective

Each perspective includes a predefined set of actions that are accessible from the menu bar and Workbench toolbar.



Save Perspective As

This command allows you to save the current perspective, creating your own custom perspective. You can open more perspectives of this type using the Window > Open Perspective > Other menu item once you have saved a perspective.



Reset Perspective

This command changes the layout of the current perspective to its original configuration.

Close Perspective

This command closes the active perspective.

Close All Perspectives

This command closes all open perspectives in the Workbench window.

Navigation

This submenu contains shortcut keys for navigating between the views, perspectives, and editors in the Workbench window.

- **Show System Menu:** Shows the menu that is used for resizing, closing or pinning the current view or editor.
- **Show View Menu:** Shows the drop down menu that is available in the toolbar of the active view.
- **Maximize active view or editor:** Causes the active part to take up the entire screen, or if it already is, returns it to its previous state.
- **Minimize active view or editor:** Causes the active part to be minimized.
- **Activate Editor:** Makes the current editor active.
- **Next Editor:** Activates the next open editor in the list of most recently used editors.
- **Previous Editor:** Activates the previous open editor in the list of most recently used editors.
- **Switch to editor:** Shows a dialog that allows switching to opened editors. Shows a dialog that allows switching to opened editors.
- **Quick switch editor:** Shows a searchable popup that allows switching to a new editor.

-
- **Next View:** Activates the next open view in the list of most recently used views.
 - **Previous View:** Activates the previous open view in the list of most recently used editors.
 - **Next Perspective:** Activates the next open perspective in the list of most recently used perspectives.
 - **Previous Perspective:** Activates the previous open perspective in the list of most recently used perspectives.

Preferences

This command allows you to indicate your preferences for using the Workbench and RT-LAB. There are a wide variety of preferences for configuring the appearance of the Workbench and its views, and for customizing the behavior of all tools that are installed in the Workbench.

Help menu

This menu provides help on using the Workbench.

Welcome

This command will open the welcome content.

Help Contents

This command displays the help contents in a help window or external browser. The help contents contains help books, topics, and information related to the Workbench and installed features.

Help Search

This command displays the help view opened on the Search page.

Dynamic Help

This command displays the help view opened to Related Topics page.

Key Assist ...

This command will display a list of key bindings.

Open Cheat Sheets ...

This command will open the cheat sheet selection dialog.

Software Updates...

This command allows you to update your product and to download and install new software.

About

This command displays information about the product, installed features, and available plug-ins.

Toolbars

There are five kinds of toolbars in the Workbench.

The main toolbar, sometimes called the Workbench toolbar, is displayed at the top of the Workbench window directly beneath the menu bar. The contents of this toolbar change based on the active perspective. Items in the toolbar might be enabled or disabled based on the state of either the active view or editor. Sections of the main toolbar can be rearranged using the mouse.

There are also individual view toolbars, which appear in the title bar of a view. Actions in a view's toolbar apply only to the view in which they appear. Some view toolbars include a Menu button, shown as an inverted triangle, that contain actions for that view.

A third type of toolbar is the perspective switcher. The perspective switcher allows quick access to perspectives that are currently open. It also has a button that can open new perspectives. The perspective switcher is normally located in the top-right, next to the main toolbar. However, it is also possible to position it below the main toolbar ("top-left"), or to position it vertically on the left-hand side of the workbench ("left"). The name of the perspectives is shown by default, but it is possible to hide the text and show only the icons. To reposition the perspective or hide the text, right-click on it and choose the appropriate item from the context menu.

Minimizing a view stack will also produce a toolbar in the trim at the outer edge of the workbench window (a Trim Stack). This bar will contain an icon for each of the views in the stack. Clicking on one of these icons will result in the view being displayed as an overlay onto the existing presentation.

Finally, the fast view bar is a toolbar that contains icons representing the current set of fast views. A fast view is a shortcut to a view that is frequently used; see the section on fast views for more information. The fast view bar appears in the bottom left corner of the workbench by default. However, it is possible to position it on the left or right as well.

In all cases, you can find out what toolbar buttons do by moving your mouse pointer over the button and reading the tooltip that opens. See the list of related reference topics below for a table of all toolbar buttons.

Related concepts

[Workbench](#)

[Views](#)

[Perspectives](#)

[Fast views](#)

Main Toolbar

The **Main Toolbar** or the Worbench Toolbar contain buttons that provide quick access to most-used commands. Generally, these commands are available in the [Workbench Menus](#) too.

Here is what the **Main Toolbar** looks like:



Figure 67:Main Toolbar overview

The different toolbar sections may be rearranged by grabbing and dragging their vertical separators. Note that sections could be added or removed by [Configuring perspectives](#) and buttons may appear or disappear when [Capabilities preference page](#) are enabled or disabled.

Toolbar icons are grayed when the corresponding commands are disallowed. Commands are enabled according to:

- the active part of the RT-LAB Workbench, it means the Editor or the View that has the focus,
- the element that is selected in this active part,
- the state of this element (for example, a running Model cannot be loaded).

The following table contains a short description for each toolbar button:

ICON	DESCRIPTION
	New Opens various wizards for resource creation.
	Save Saves the resource currently opened in the active editor.
	Build Build (compile) a Model or manages build configurations.
	Assign Opens the Assignment page of a Model editor
	Load Loads a Model.
	Execute Starts the execution of a Model.
	Pause Pause the execution of a Model.
	Reset Stops the execution of a Model.
	Snapshot Takes and restores a model snapshot and load and save parameters. See Taking a Snapshot, Load Parameters wizard and Save Parameters wizard for more help.
	Matlab Opens Matlab. Specify the version to override the default version.
	ScopeView Launches the ScopeView application to acquire and display signals from real-time simulation. See Using ScopeView for more information.

ICON	DESCRIPTION
	Probe Control Opens the Probe Control Panel .
	Python Run a Python script, opens the Interactive Python Console or use the Macro recorder.
	Python Debug a Python script.
	Search Opens the Search View .
	Next Annotation Go to the next annotation in the current editor. Use submenu to select annotation types to be considered (errors, bookmarks, search results...).
	Previous Annotation Go to the previous annotation in the current editor. Use submenu to select annotation types to be considered (errors, bookmarks, search results...).
	Last Edit Location Shows the last change that occurred within a file. Opens an editor if necessary. Hit this button several times to go back to older edit locations.

Wizards

Contents

- [New RT-LAB project wizard](#)
- [New RT-LAB model wizard](#)
- [Existing RT-LAB model import wizard](#)
- [Add Model wizard](#)
- [New Target wizard](#)
- [Detected Targets Wizard](#)
- [New Folder wizard](#)
- [New File wizard](#)
- [Flash bitstream wizard](#)
- [Connect to Embedded Simulation Wizard](#)
- [Load Parameters wizard](#)
- [Save Parameters wizard](#)

New RT-LAB project wizard

Main interface panel

Use **File > New > RT-LAB Project** to open the dialog that enables you to create a new project in the workbench.

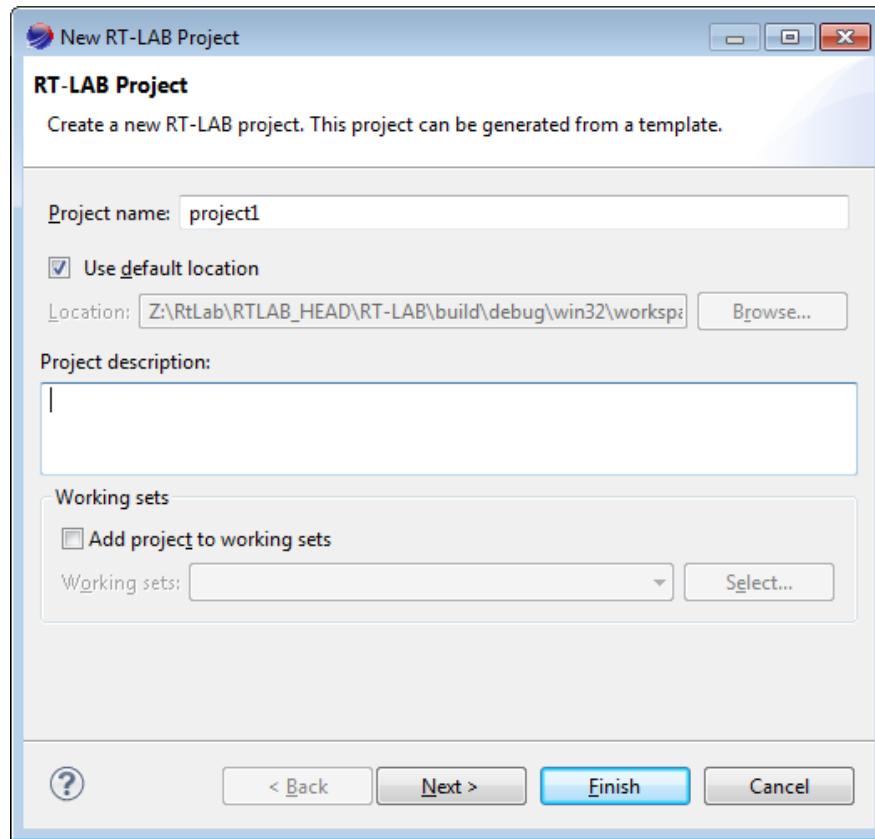


Figure 68:New Project wizard: main page

Description

Each simulated models have to belong to one project. See [Project Explorer](#) for a detailed description of **Project** and other related concepts.

This dialog explains how to created a new empty project or project that contains a model.

Parameters

The first page of the wizard helps you to set the parameters of the project.

Project name: The name of the new project to be created. This field is required to create a new project. A project name must be unique within the current workspace.

Use default location: The default location where RT-LAB will store your project is in the \$workspaceDirectory\$ directory. De-select "Use default location" to specify a location other than the default. You can type the new location or browse to select a file system location. Before changing the location of a project, please read [Organizing resources](#) in the [Project Explorer](#) documentation.

Project description: A short description to be associated to the new project.

Creating a new empty project

Once valid parameters have been entered, an empty project can be created by clicking on the <Finish> button on the first page of the wizard. Then it will be possible to add new or existing models to this project, using respectively the **New RT-LAB model wizard** or the Add model wizard.

Creating a new project from a template

Clicking on the <Next> button will display a list of available project templates. Each template illustrates an RT-LAB feature and provides at least one configured model. Other files may be included, such as help files, python scripts, test sequences, C code...

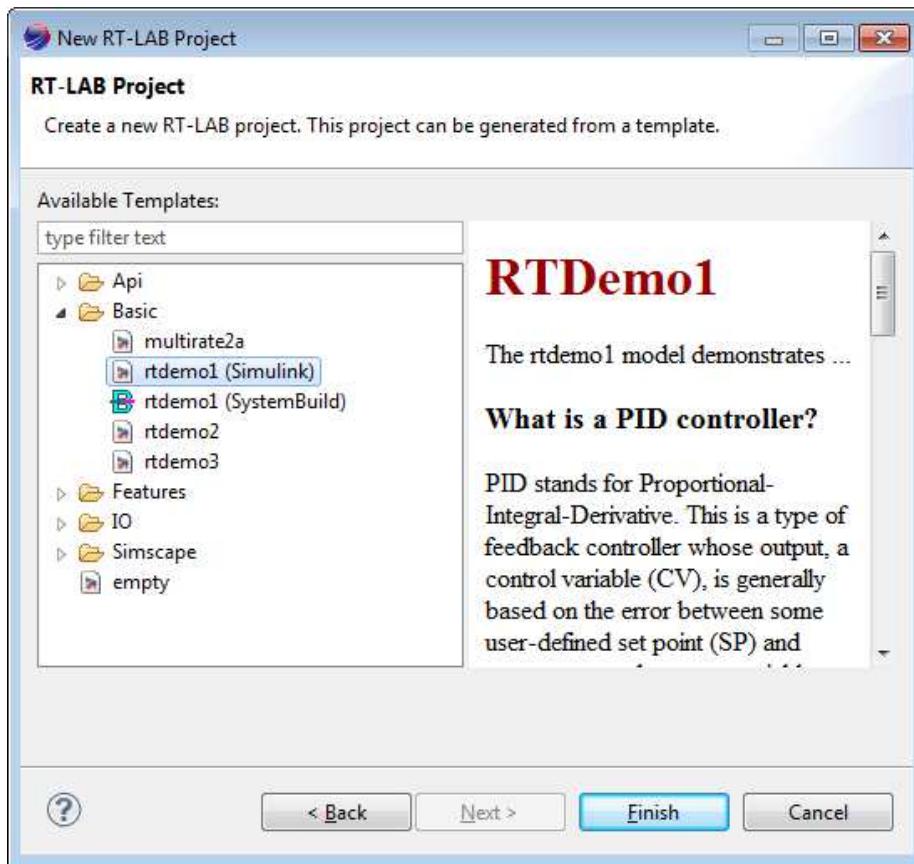


Figure 69:New Project wizard: selecting a template

The left part of the dialog is a template explorer.

The right part of the dialog contains the description of the currently selected template.

Templates can also be filtered to quickly find all the templates associated to a particular feature:

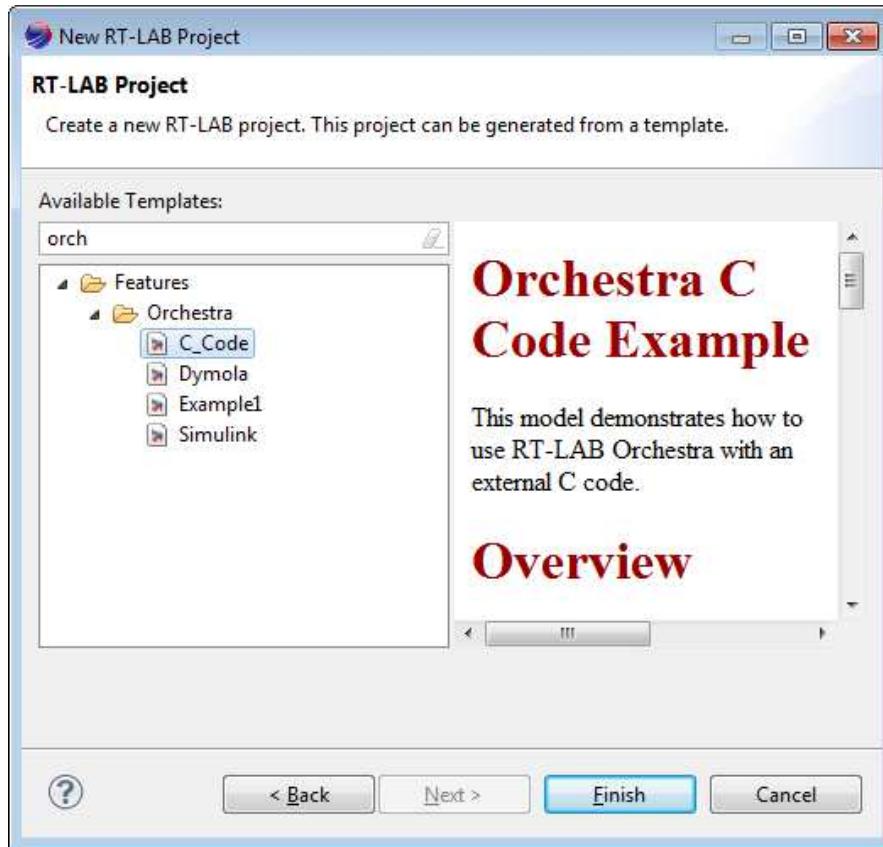


Figure 70:New Project wizard: filtering templates

Clicking on <Finish> will generate a new project from the selected template. This project will then appear in the **Project Explorer** and will be ready to begin a simulation.

Note that modifying files in the newly created project will not affect the template itself. This allows to make lots of changes and to create another similar and clean project if things go wrong.

New RT-LAB model wizard

Main interface panel

Use **File > New > RT-LAB Model** to open the dialog that enables you to create a new model in the workbench.

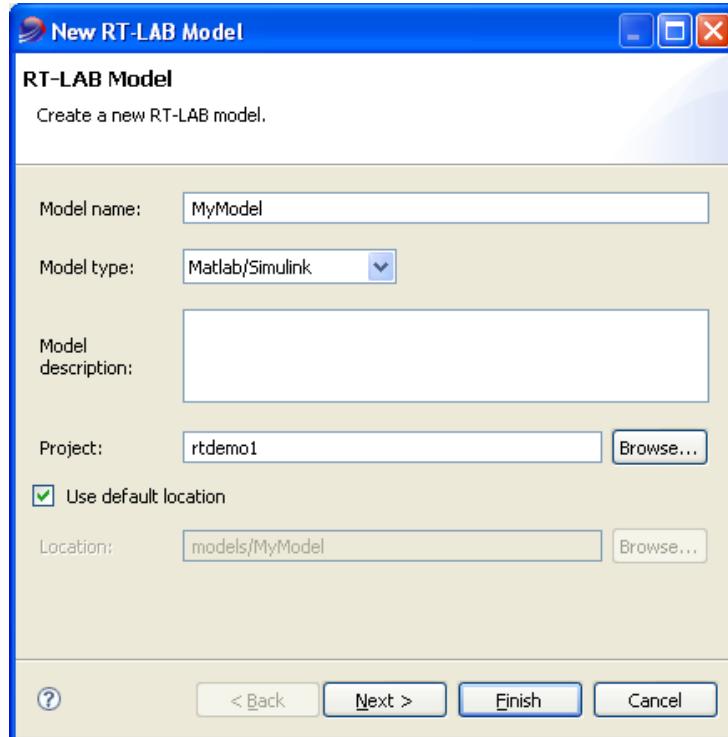


Figure 71:New Model wizard: main page

Description

This wizard explains how to create a new empty model or a model that is a copy of one supplied template model.

Parameter

The first page of the wizard helps you to set parameters of your model.

Model name: Name of the new model. This field is required.

Model type: Matlab/Simulink or EMTP-RT.

Model description: Description that the user wants to associate to the model.

Project: Project that will contain the new model. This field is required.

Note: the user has to create a new project if there is no existing one.

Use default location: The default location where RT-LAB will store your model is in the \$projectDirectory\$/models/modelName directory. De-select "Use default location" to specify a location other than the default. You can type the new location or browse to select a file system location.

Before changing the location of a model, please read [Organizing resources](#) in the [Project Explorer](#) documentation.

Creating a new empty model

After setting the parameters's values of the new RT-LAB model wizard, the user could create an empty model by clicking on the <Finish> button on the first page of the wizard. An empty model is a diagram with no functional elements as blocks or connection lines.

Creating a new model based on one template model

After setting the parameters's values of the new RT-LAB model wizard, the user could also create a model based on one template model by clicking on the <Next> button.

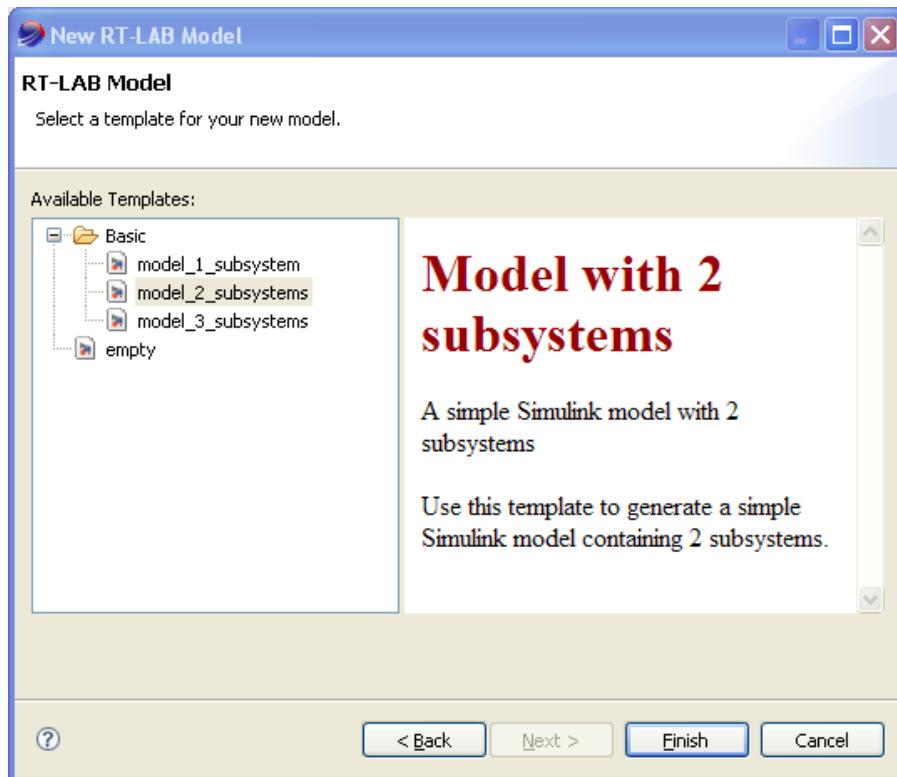


Figure 72:New Model wizard: selecting a template

The left part of the dialog is a template explorer.

The right part of the dialog contains the description of the currently selected template.

Click on the <Finish> button to create a new model from the selected template. This model will be automatically added to the project specified on the main page.

Note that modifying the newly created model will not affect the template itself.

Import wizard

This wizard helps you import resources into the Workbench.

When the Import wizard first comes up, you must choose what type of import to do. To assist in locating a particular wizard, the text field can be used to show only the wizards that match the entered text.

Archive File

If you choose this option, you will import files from an archive file.

Archive File

The file from which to import. Type in the full path or Browse to select the path on the file system.

Filter Types...

Dialog to select which file types to import. Use this to restrict the import to only certain file types.

Select All

Check off all resources for import

Deselect All

Uncheck all resources.

Folder

The folder into which the resources will be imported. Type the path or Browse to select a path in the Workbench. The folder holding the selected resource

Overwrite existing resources without warning

Determines whether importing a resource should silently overwrite a resource which already exists in the Workbench. If this option is off, you will be prompted before a given resource is overwritten, in which case you can either overwrite the resource, skip it, or cancel the import. Off

Existing Project into Workspace

Imports a project into this workspace that was previously located in this workspace, or that currently exists in another workspace.

Select root directory

Root directory in the File System to start scanning for projects to import. Type in the full path or Browse to select the path on the file system.

Select archive file

Archive file to scan for projects to import. Type in the full path or Browse to select the archive on the file system.

Select All

Check all of the projects that were found for import.

Deselect All

Uncheck all projects.

Refresh

Rescan the selected source for projects to import.

Copy projects into workspace

When selected this will cause the imported project to be copied into the current workspace.

Import as binary project

File System

If you choose this option, you will import files from the file system.

Directory

The directory from which to import files. Select a previous path from the drop down combo or Browse to select the path in the file system.

Filter Types

Dialog to select which file types to import. Use this to restrict the import to only certain file types.

Select All

Check off all files and folders for import.

Deselect All

Uncheck all resources.

Folder

The folder into which the resources will be imported. Type the path or Browse to select a path in the Workbench. The folder holding the selected resource

Overwrite existing resources without warning

Determines whether importing a resource should silently overwrite a resource which already exists in the Workbench. If this option is off, you will be prompted before a given resource is overwritten, in which case you can either overwrite the resource, skip it, or cancel the import. Off

Create complete folder structure

Create hierarchy (folder) structure in the Workbench to accommodate the resources being imported, and all parent folders of those resources in the file system.

Create selected folders only

Create hierarchy (folder) structure in the Workbench to accommodate the resources being imported.

Preferences

Import preferences from the local file system.

From preference file

The file from which to import preferences. Select a previous file from the drop down combo or Browse to select the file in the file system.

Import All

Import all of the preferences.

Choose specific preferences to import

Choose from the preferences contained in the file, like CVS connection preferences or JRE preferences.

Select All

Check off all files and folders for import.

Deselect All

Uncheck all resources.

Add Model wizard

This wizard explains how to add an existing RT-LAB model to a project. The model will be added to the selected destination project inside the Models folder.

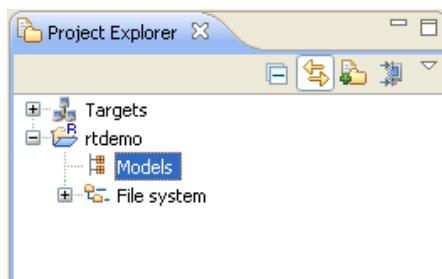


Figure 73:Models folder

The next sections show how to add an existing RT-LAB model to a project.

By default, the **Project Explorer** view is included in the Edition perspective. To add it to the current perspective, click Window > Show view > Other... > General > Project Explorer

How to add an existing RT-LAB model

To add an existing model, click on the Add > Existing Model menu item. Right click on a project or its Models folder to open the contextual menu.

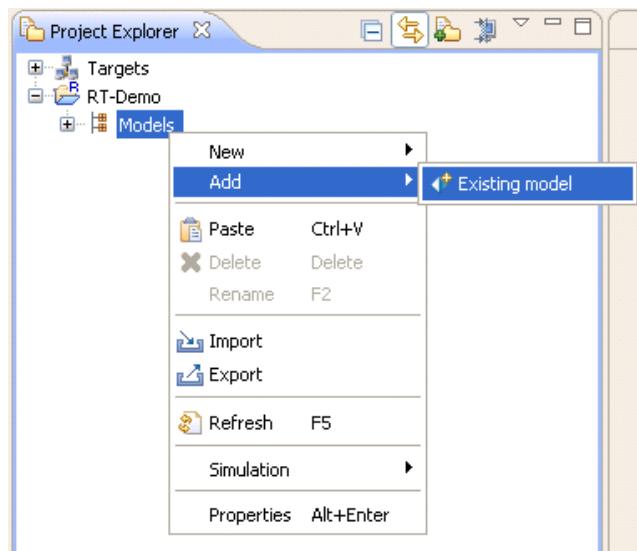


Figure 74:Add existing model menu

This operation has opened the Add Model dialog:

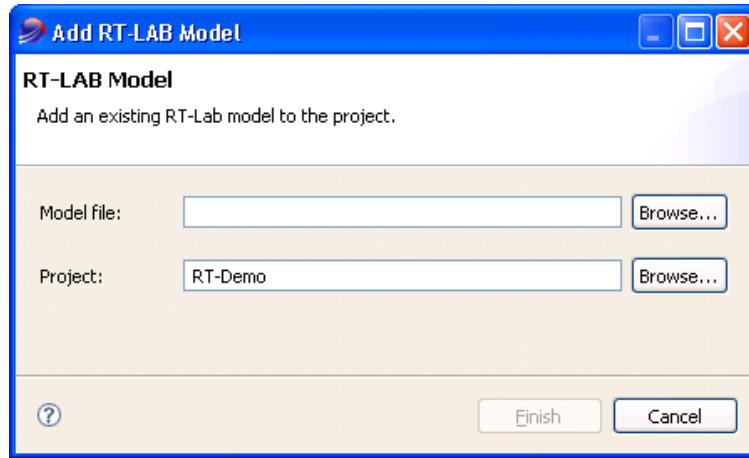


Figure 75: Add model dialog

Then you can browse the file system to select any model file you want to add to the selected project. This project should already be selected in the 'Project' field. However, click on the 'Browse...' button next to it to select another project from the workspace.

If the model file is located outside of the project folder, a linked resource (see [Other resources](#)) will be created so that the model file can be accessed from the [Project Explorer](#):

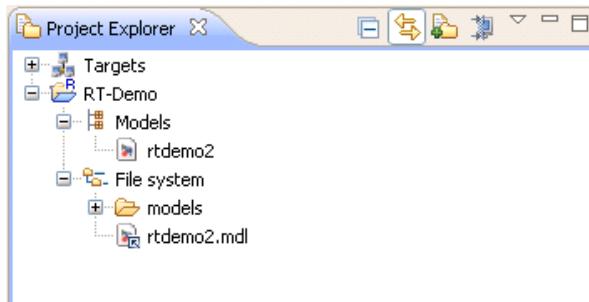


Figure 76: Add Model result (linked resource)

Important notes

Once added to a project, a model can be edited like any other model. For that reason, example models provided by RT-LAB should never be added to a project using this wizard: these models must not be modified since they are used as templates when creating new projects. Prefer using the [New RT-LAB project wizard](#) or the [Existing RT-LAB model import wizard](#) when you want to use an example model.

A faster way to add an existing model to a project is to use [Drag and Drop](#). Simply drag a model file from anywhere on your file system and drop it on a Models folder. Hold the <Ctrl> key to make a copy of the model file before adding it to the Models folder.

Related concepts

- [Views](#)
- [Perspectives](#)

Related tasks

[Opening views](#)

[Moving and docking views](#)

Related reference

[Project Explorer](#)

[Existing RT-LAB model import wizard](#)

New Target wizard

This wizard explains how to create a new target to a local or remote computer. The new target will be added to the current list of targets in RT-LAB. This list is available in the **Project Explorer** view just above the projects.

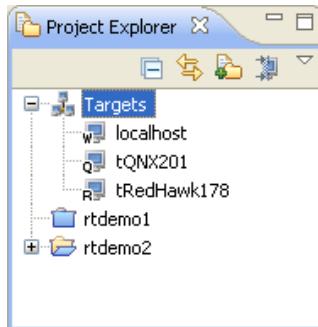


Figure 77:List of targets in the Project Explorer view

RT-LAB supports different target platforms: Windows, QNX, Redhat and Redhawk. See [Target platform](#) for more information.

The next sections show how to create a new target and give some details about the properties of a target.

By default, the **Project Explorer** view is included in the Edition perspective. To add it to the current perspective, click Window > Show view > Other... > General > Project Explorer

How to create a new target

To create a new target, select the New Target menu item from the menu New of the Targets. Right click on the Targets in the Project explorer view to open the contextual menu of Targets.



Figure 78:New Target menu item

This operation has opened the New RT-LAB Target dialog. Specify the name and the IP Address of the target. (Note: Ask your network administrator for the target ip address). To create a target on the local computer, enter the ip address: 127.0.0.1

Press the Ping button to make sure that the target is available and to verify if the target is configured properly with the good software version.

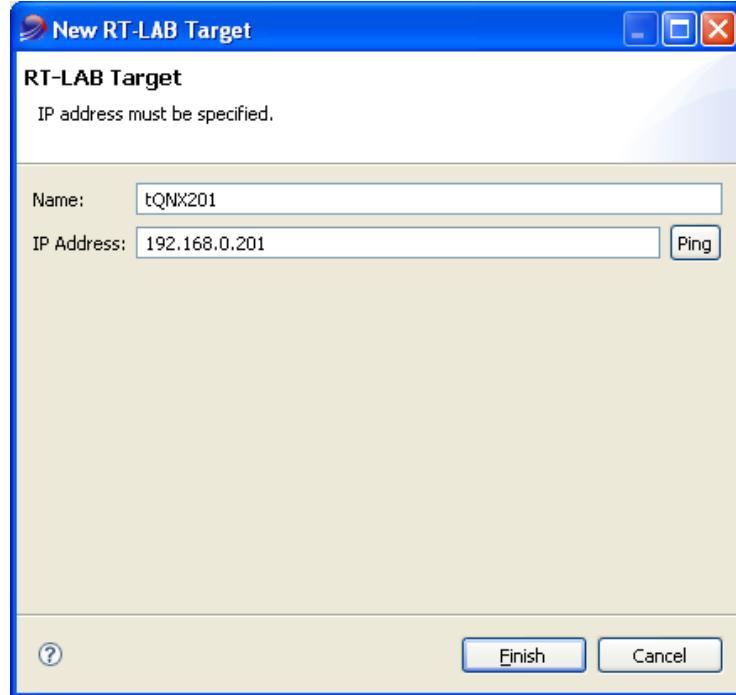


Figure 79:New RT-LAB Target

The next figure shows the result of a positive ping.

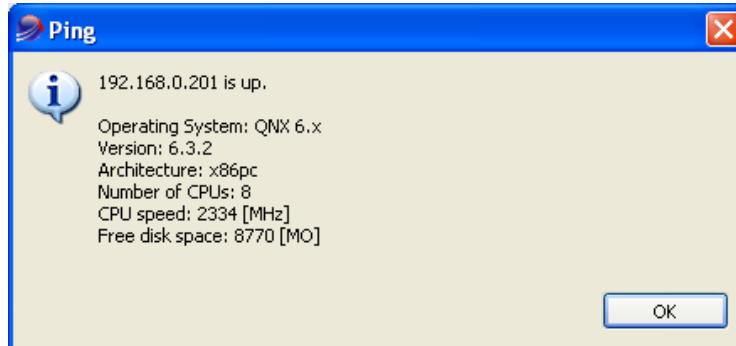


Figure 80:Positive ping result

Press OK button to close the ping result and press Finish button in the New RT-LAB Target dialog to complete the configuration of the target. Now, the target has been created and added to the targets list.

If the ping has not passed, you will receive this message:



Figure 81:Negative ping result

Verify the possible causes that are display in the message. Make sure that the ip address of the target is the good one. For more information, see the RT-LAB Installation Guide.

Properties of a target

Each target in the list of targets has their own properties. To consult these properties, right click on the specific target and click on the Properties menu item. The Properties view displays the name, the ip address, the operating system and the hardware of the target. The Properties view will be added in a tabbed notebook at the rightmost position. See the next figure to know which information is provided by the Properties view.

Properties	
1 items selected	
Advanced	
Property	Value
1- Overview	
IP address	192.168.0.201
name	tQNX201
2- Operating System	
detected platform	QNX 6.x
version	6.3.2
3- Hardware	
architecture	x86pc
CPU speed [MHz]	2333
disk space [MO]	8812
number of CPUs	8

Figure 82:Target Properties view

Related concepts

- [Views](#)
- [Perspectives](#)
- [Toolbars](#)

Related tasks

- [Opening views](#)
- [Moving and docking views](#)

Related reference

[Target platform](#)
[Project Explorer](#)

Detected Targets Wizard

This wizard is used to manage newly detected RT-LAB Targets. Each detected Target can be either added to the workbench or ignored.

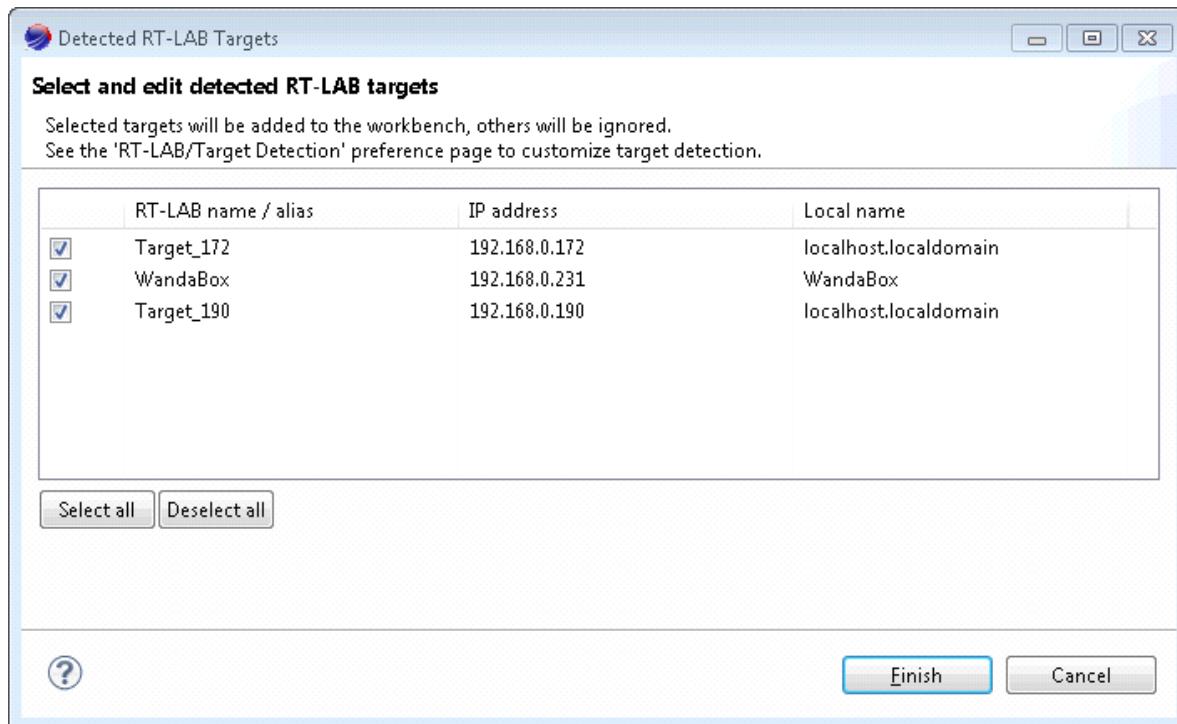


Figure 83: Detected Targets Wizard

How to open this wizard

This wizard can be opened in three different ways:

- Clicking on a "New Target" notification

If target notifications are enabled (see [Target Detection preference page](#)), a popup will be displayed each time a new target is detected on the network:

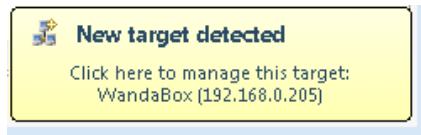


Figure 84: "New Target" notification

Clicking on this notification will open the Detected Targets Wizard.

- Clicking on the "New Target" icon on the status bar

When new targets are available, an icon is displayed on the Status Bar:



Figure 85: Status bar icon of the Detected Targets Wizard

Clicking on this icon will open the Detected Targets Wizard.

- Performing a “Target discovery” from the [Project Explorer](#)

At any time, even if automatic target detection is disabled, you can start a target detection over the network: right-click on the “Targets” item of the Project Explorer and select “Discover targets”:

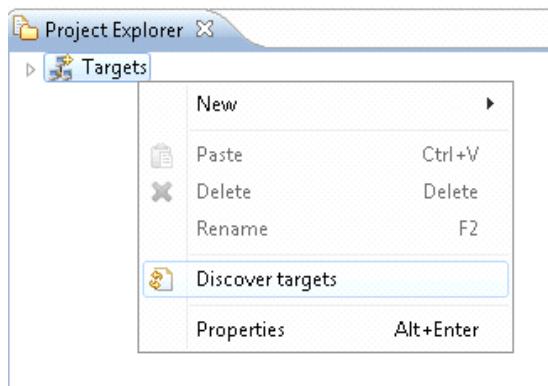


Figure 86: Discovering targets over the network

This will enable the target detection for a few seconds and if any new target is detected, the Detected Targets Wizard will be opened.

Naming Targets

Each RT-LAB Target of the workbench must have a unique name. This is an alias that is saved on the Host computer

A Target has also a “local name”, stored on the target itself and shared with all users. By default, the hostname (defined by the Target’s Operating System) is used. This local name appears in the Detected Targets Wizard and can be changed later using the Target Editor. It is stored in a text file located at /usr/opalrt/local/targetname.

If the local name is a default one (such as “localhost”) or if this name already exists in the Workbench, the Detected Targets Wizard automatically generates a unique name such as “Target_XXX” where XXX are the last 3 digits of the Target’s IP address.

Hidden targets

If you set the local name of a Target to “hidden”, this Target will not be detected by this wizard anymore so other users will not be notified when this Target is connected to the network. However, this Target still remains accessible and can be added to any workbench by using the [New Target wizard](#).

Ignoring targets

When pressing "Finish", unselected targets will be ignored, at least until RT-LAB is restarted or until a target discovery is performed. Optionally, targets may be definitively ignored by clicking "Yes" when prompted:

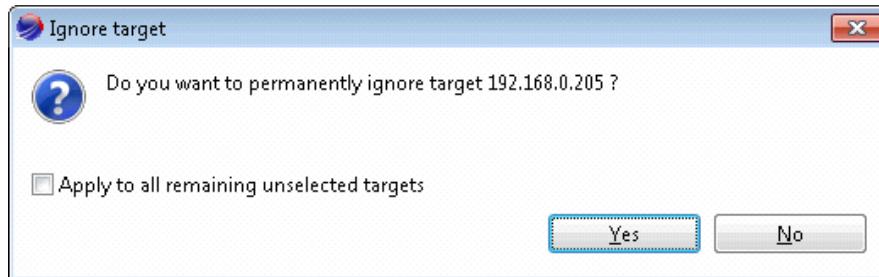


Figure 87:Ignoring detected targets

The list of ignored targets can be seen and edited in the [Target Detection preference page](#).

This preference page also allows to limit the target detection to specific detection ranges.

Related concepts

[Wizards](#)

Related reference

[Target Detection preference page](#)

[New Target wizard](#)

[Project Explorer](#)

New Resource Project wizard

This wizard helps you create a new resource project in the Workbench. This project is a generic project without any specific functionalities, so it should not be used to hold a simulation project (see [New RT-LAB project wizard](#) for more information).

Parameters

Project Name

The name of the new project to be created.

Location

The location in the file system where the project will be created. De-select **Use default location** to specify a location other than the default. You can type the new location or browse to select a file system location for the new project.

Select Referenced Projects page

In the Referenced Projects list, you can set project dependencies for the new project. In the list of other projects in the Workbench, you can select one or more projects on which you want the new project to depend. Initially, no projects will be selected.

Related Concepts

[New RT-LAB project wizard](#)

[Project Explorer](#)

New Folder wizard

This wizard helps you create a new folder in the Workbench.

Here is what the New Folder wizard looks like:



Parameters

Enter or select the parent folder

The resource in which the new folder will be created. Type or navigate the list to select the resource. The default value is the resource that was selected when you chose to create the new folder.

Folder name

The name for the new folder.

Advanced

The Advanced button reveals or hides a section of the wizard used to create a linked folder. Check the **Link to folder in the file system** checkbox if you want the new folder to reference a folder in the file system. Use the field below the checkbox to enter a folder path or the name of a path variable. Use the **Browse...** button to browse for a folder in the file system. Use the **Variables...** button if you want to use a path variable to reference a file system folder.

Related Concepts

[Linked resources](#)

New File wizard

This wizard helps you create a new file in the Workbench.

Here is what the New File wizard looks like:



Parameters

Enter or select the parent folder

The resource in which the new file will be created. Type or browse the list to select the resource. The default value is the resource that was selected when you invoked the New File wizard.

File name

The name for the new file, including the file extension.

Advanced

The **Advanced** button reveals or hides a section of the wizard used to create a linked file. Check the **Link to file in the file system** checkbox if you want the new file to reference a file in the file system. Use the field below the checkbox to enter a file path or the name of a path variable. Use the **Browse...** button to browse for a file in the file system. Use the **Variables...** button if you want to use a path variable to reference a file system file.

Related Concepts

[Linked resources](#)

Existing RT-LAB model import wizard

This wizard explains how to import an existing RT-LAB model to a project. The model will be added to the select destination project inside the Models folder.

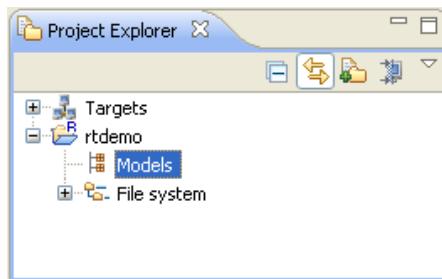


Figure 88:Models folder

The next sections show how to import an existing RT-LAB model to a project.

By default, the **Project Explorer** view is included in the Edition perspective. To add it to the current perspective, click Window > Show view > Other... > General > Project Explorer

How to import an existing RT-LAB model

To import an existing model, click on the Import... menu item. Right click on the project destination folder to open the contextual menu.

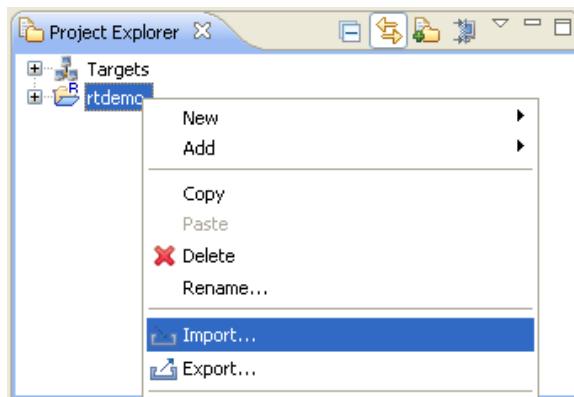


Figure 89:Import menu

This operation has opened the Import dialog. In this dialog, select the "Existing RT-LAB model" from the folder RT-LAB. Click Next.

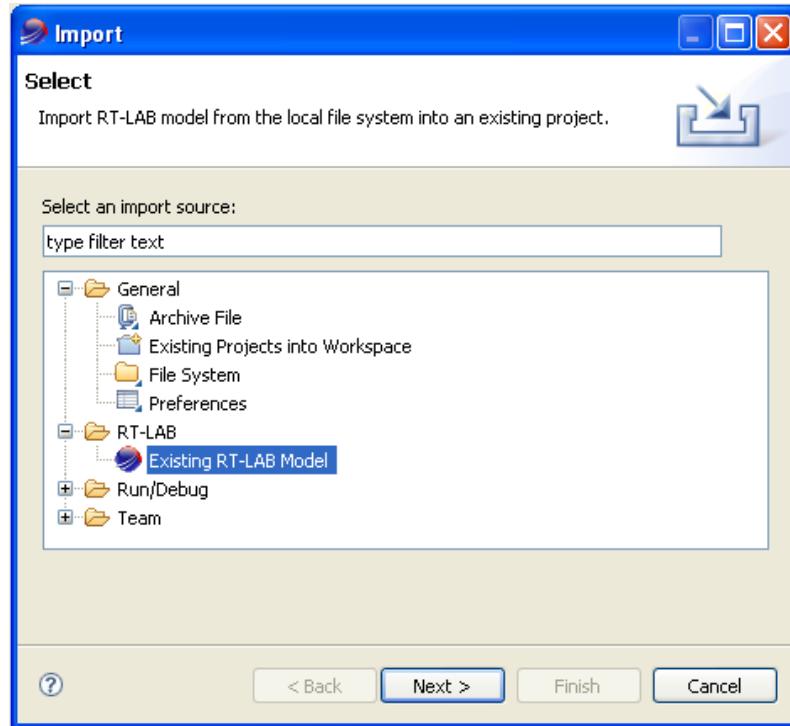


Figure 90:Import dialog

The import existing RT-LAB model step is show. To quickly import the model, click on the Browse button to find the model's file to add to the project. Verify if the project field is the desired destination folders. Click the Finish button to import the files to the project folder.

Also, it is possible to change the default location by browsing the destination folder. The location is the folder where the model's file will be added.

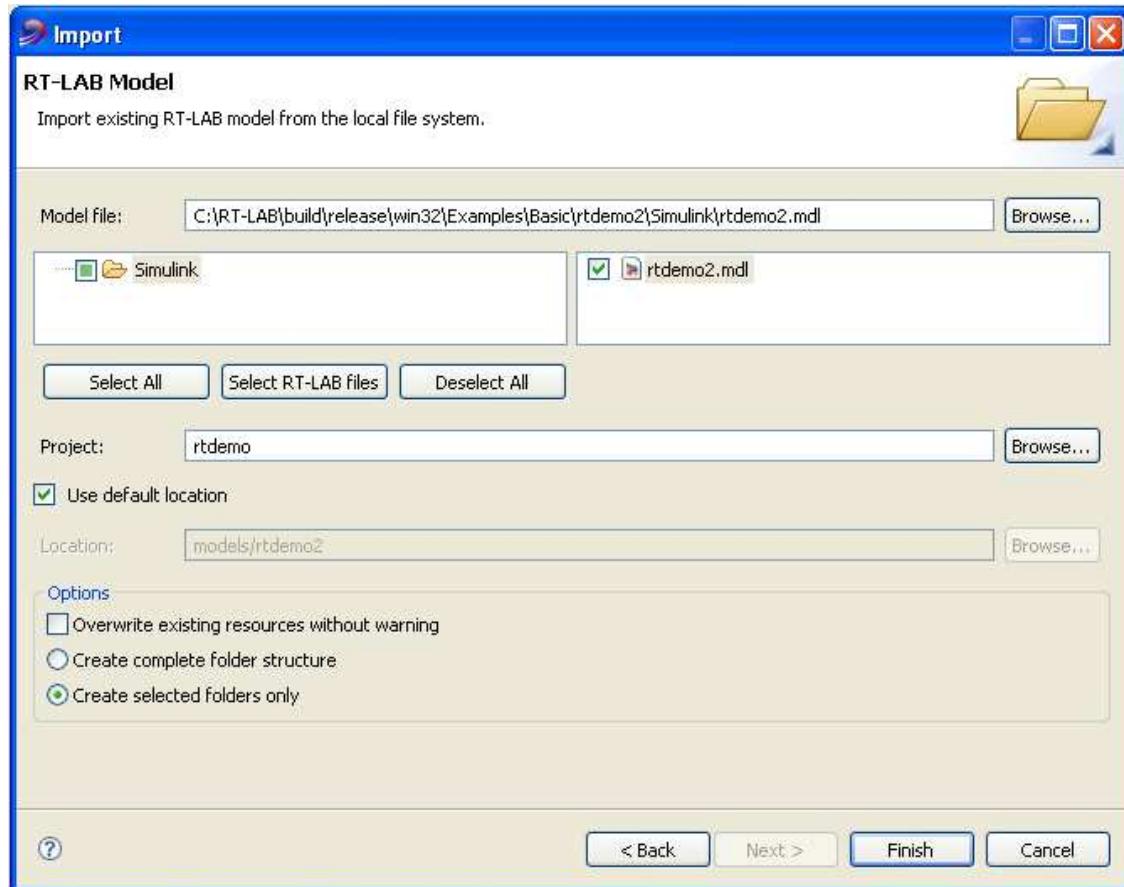


Figure 91:Import model dialog

Import model dialog options

Option Overwrite existing resources without warning is used to force the selected files to be imported in the project folder. Files will not be imported if they already exist in the RT-LAB project folder. If we check this option, the existing files in the project folder will be overwritten with the new one.

Option Create complete folder structure is used to create the source folder structure to the destination folder of the project. In this example the created destination structure is: models/rtdemo2/RT-LAB/build/release/win32/Examples/Basic/rtdemo2/Simulink/rtdemo2.mdl

Option Create selected folders only is used to create only the selected folder to the destination folder of the project. In this example the created destination structure is: models/rtdemo2/rtdemo2.mdl

Related concepts

[Views](#)
[Perspectives](#)

Related tasks

[Opening views](#)
[Moving and docking views](#)

Related reference

[Project Explorer](#)

Export wizard

This wizard help you export resources from the Workbench.

When the Export wizard first comes up, you must choose what type of export to do. To assist in locating a particular wizard, the text field can be used to show only the wizards that match the entered

Archive File

If you choose this option, you will export files to an archive file.

Select resources to export

The project (and resources within that project) to export to an archive.

Select Types...

Dialog to select which file types to export. Use this to restrict the export to only certain file types.

Select All

Check off all resources for export.

Deselect All

Uncheck all resources.

Archive File

The path and name of an archive file into which the resources will be exported. Type the path, select a previous path from the drop down list, or Browse to select a path and file name on the file system. The archive file of the previous export, or <blank>.

Zip file

Export the file in zip format true

Tar file

Export the file in tar format true

Compress the contents of the file

Compresses the contents (resources selected to be exported) in the archive that is created. On

Overwrite existing file without warning

If the specified archive already exists in the file system, you will be prompted to overwrite the file. If you do not want to be prompted turn this option on. Off

Create directory structure for files

Create hierarchy (folder) structure in the file system as it exists in the Workbench.

Create only selected directories

Create hierarchy (folder) structure in the file system only for selected folders.

File System

If you choose this option, you will export files to the file system.

Select resources to export

The project (and resources within that project) to export to the file system.

Select Types...

Dialog to select which file types to export. Use this to restrict the export to only certain file types.

Select All

Checks off all resources for export.

Deselect All

Uncheck all resources.

Directory

The directory on the file system into which the resources will be exported. Type the path, select a previous export path from the drop down list, or Browse to select a path.

Overwrite existing files without warning

Determines whether exporting a resource should silently overwrite a resource which already exists in the file system. If this option is off, you will be prompted before a given file is overwritten, in which case you can either overwrite the file, skip it, or cancel the export.

Create directory structure for files

Create hierarchy (folder) structure in the file system as it exists in the Workbench.

Create only selected directories

Create hierarchy (folder) structure in the file system only for selected folders.

Preferences

Export preferences to the local file system.

Export All

Export all of the preferences in this session.

Choose specific preferences to export

Select preferences from this session to export, like CVS repository preferences.

Select All

Select all of the available preferences.

Deselect All

Clear all of the available preferences.

To preference file

A file on the file system to store the preferences. Type the file, select a previous export file from the drop down list, or Browse to select a file.

Overwrite existing files without warning

Overwrite a pre-existing file.

Flash bitstream wizard

The **Select a bitstream** wizard allows to flash an I/O board with any compatible bitstream.

This wizard is available from the **Target** context menu: **Execute > Flash bitstream** (this menu is not available for Windows Targets).

Here is what the wizard looks like:

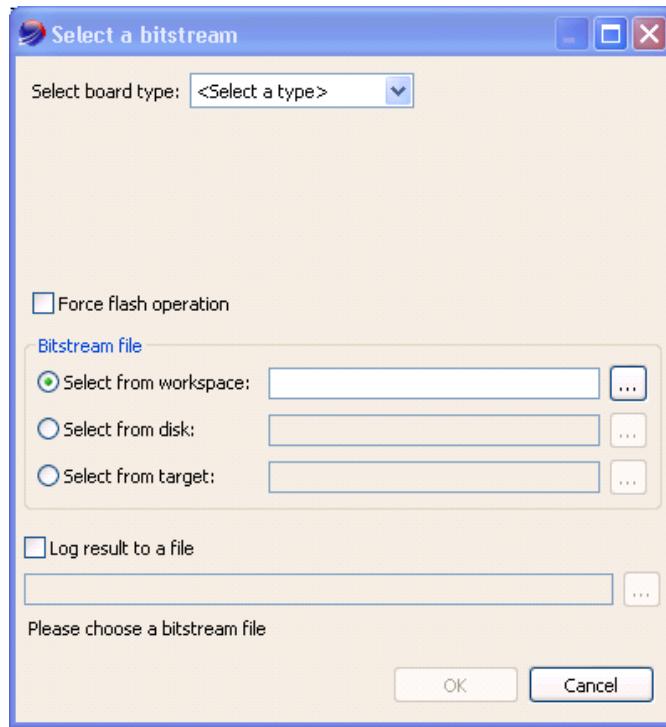


Figure 92:Flash bitstream wizard (overview)

Follow these steps to flash a board:

- Select the board type. For example, OP5142 or TestDrive - SP3.
- Specify corresponding options to identify a particular board if several boards of the same type are present in the **Target**. For example: specify the slot ID of a TestDrive module. If no option is specified, RT-LAB will flash the first board that it finds.
- Select the bitstream file to be used. The "browse" dialog will only show compatible files.

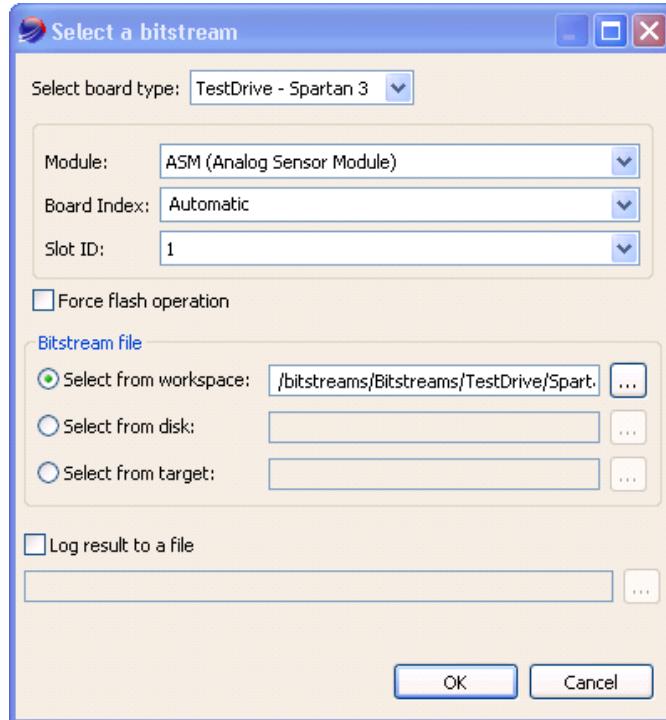


Figure 93: Flash bitstream wizard (example)

Alternatively, you can browse first for any bitstream file. In this case, the board type will be detected and automatically selected. Then you can adjust options, such as board ID or slot ID to flash a particular board.

Similarly, the **Flash bitstream wizard** wizard can be opened by dragging and dropping a bitstream file from the workspace or the file system to any **Target** of the **Project Explorer**.

The bitstream file may be located in the workspace, on the local file system or on the Target. If necessary, the selected file will automatically be transferred to the **Target** and will be available in the folder /home/ntuser/bitstreams/.

If the "Force flash operation" is selected, the selected board will be flashed even if it is already programmed with the specified bitstream.

Once both board and compatible bitstream have been selected, you can click on the "OK" button to launch the process. During the flash operation, results will be shown in the **Target Console** and a popup dialog will be displayed :

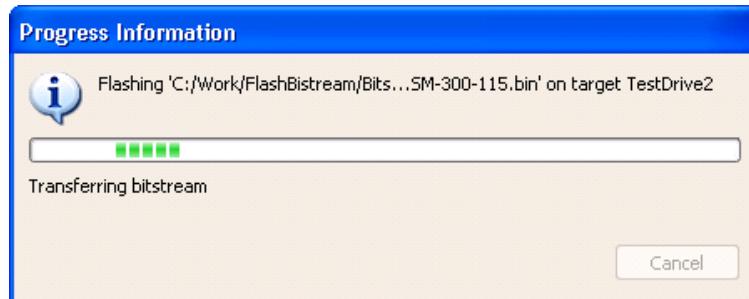
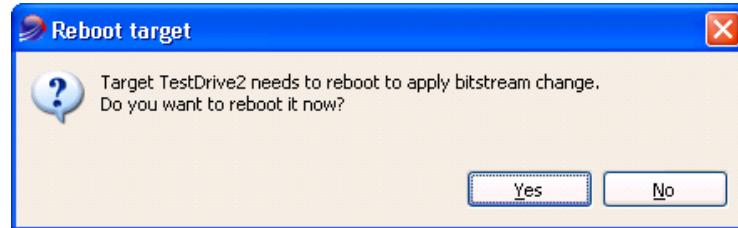


Figure 94: Flashing bitstream: popup dialog

Results can be saved to a file by selecting the "Log result to a file" option.

For certain boards, like OP5110, you have to reboot the **Target** to put the bitsream change into effect.
In this case, a dialog will be displayed:



Connect to Embedded Simulation Wizard

This wizard allows users to connect an existing RT-LAB project to an embedded simulation. The project can then be opened to interact with the embedded simulation. To learn more about embedded simulation in general, see [Embedding Simulation](#).

Embedded simulations are shown as children of RT-LAB targets in the **Project Explorer**. Targets that run an embedded simulation and projects that are connected to an embedded simulation are identified by having a short description such as [Embedded on *targetName*] appended to their name:

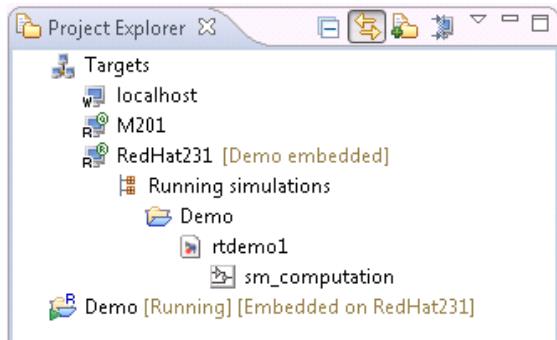


Figure 95: Embedded Simulation

The next sections show how to connect an existing RT-LAB project to an embedded simulation.

By default, the **Project Explorer** view is included in the Edition Perspective. To add it to the current Perspective, click Window > Show view > Other... > General > Project Explorer

Connecting a Project to an Embedded Simulation

To connect a project to an embedded simulation, right click on the Running Simulation and select Connect.

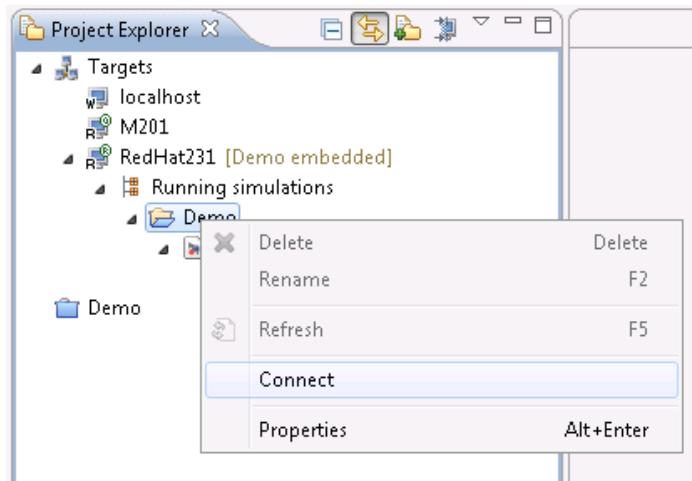


Figure 96: Embedded simulation menu

This operation has opened the Embedded Simulation dialog.

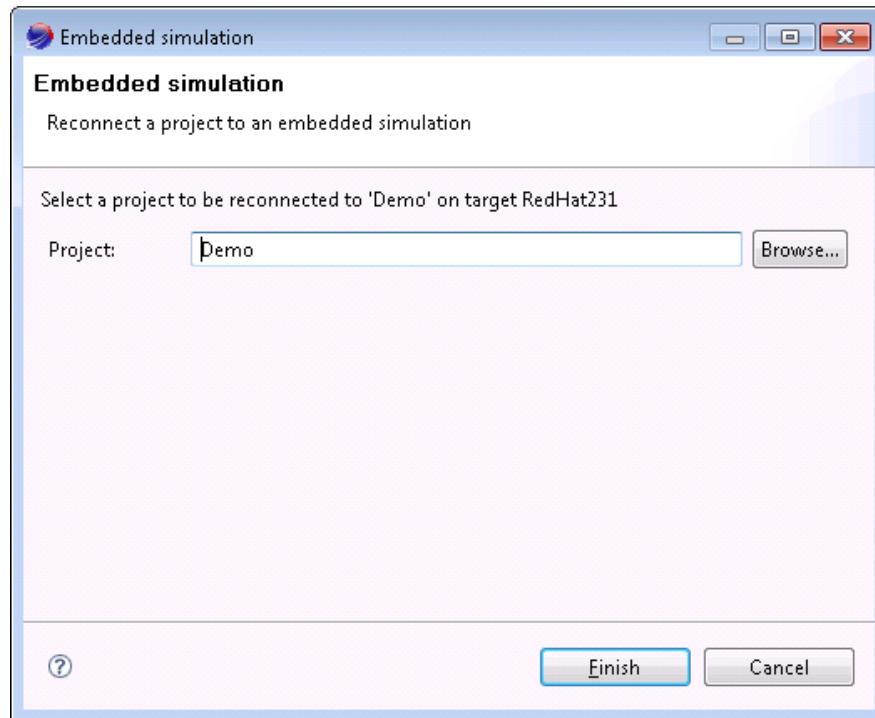


Figure 97: Embedded simulation dialog

If the project that was used to start the simulation exists in the workbench, it should already be selected in the 'Project' field.

Otherwise, clicking on the 'Browse...' button next to it allows you to select another project from the workspace. However, be careful when choosing another project: it must contain an exact copy of the currently embedded model. Trying to connect to an embedded model with a different model may cause the simulation to crash.

The selected project must be closed and must neither be running nor active to enable the reconnection. See detailed information on project states in [Model and project states](#).

Click Finish to start the reconnection:

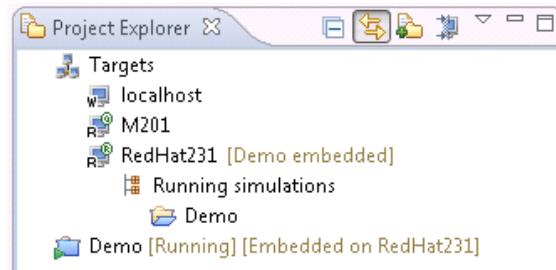


Figure 98: Project is ready to be connected

Then you can simply open the project and use it as usual to interact with the simulation.

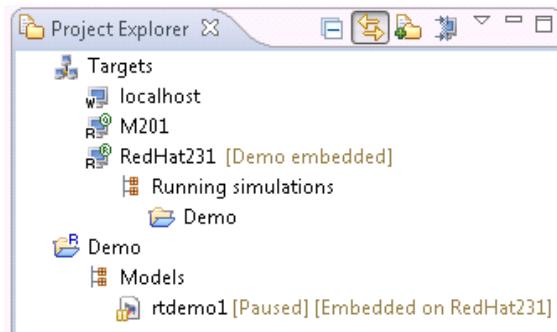


Figure 99:Project is connected

If you close the project, it will automatically be disconnected from the embedded simulation. Follow this procedure again to reconnect it.

Related concepts

[Views](#)
[Perspectives](#)

Related tasks

[Embedding Simulation](#)
[Opening views](#)
[Moving and docking views](#)

Related reference

[Project Explorer](#)

Load Parameters wizard

This wizard helps you load the values of a set of parameters and apply them to the model currently selected in the **Project Explorer**.

Selecting the Parameters File

The first wizard page lets you select the file that you want to load. Here is what the page looks like:

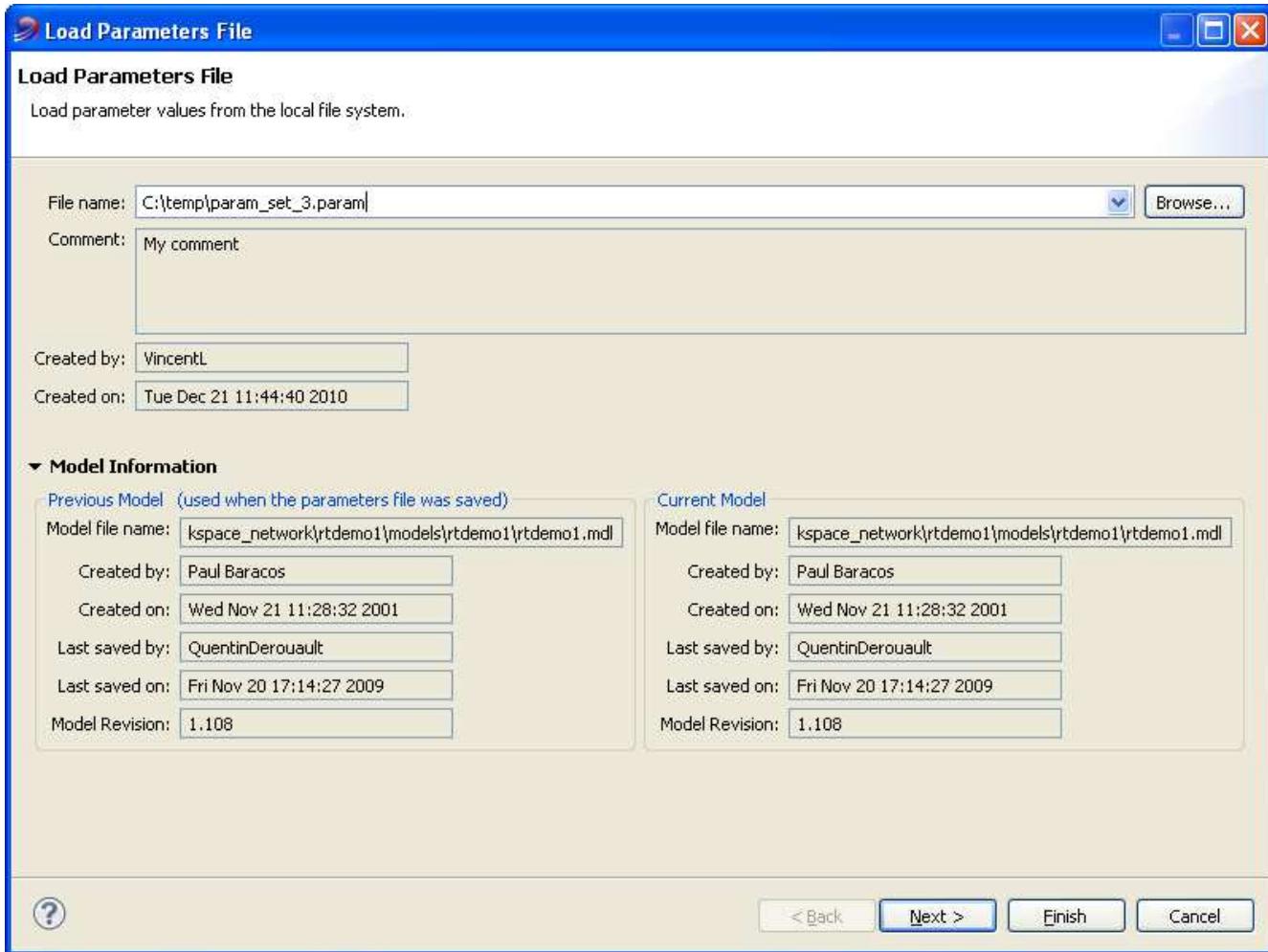


Figure 100:Load parameters dialog

When you select a valid file, the wizard will compare the structural content of the previous model, i.e. the one that was used to save the file, to the content of the current model. If it detects structural differences, a warning will be displayed indicating that models are different and that some parameters may have been added, removed, renamed or moved since the writing of the parameter file. The previous and current model information will also be shown at the bottom of the page allowing you to compare the history and revision of each model. See **Solving Structural Differences** page for more help.

When no differences are detected or all structural differences are resolved, the last page of this wizard will help you to view, compare and edit the loaded value. See **Viewing, Comparing and Editing Values** page for more help.

Here is the description of the fields of the first page:

Filename: File path and name (including .param extension) where the file is located. Use the dropdown arrow to select a file from the history or the browse button to select a file from the file system.

Comment : Comment stored with your parameters.

Created by : Parameter file author.

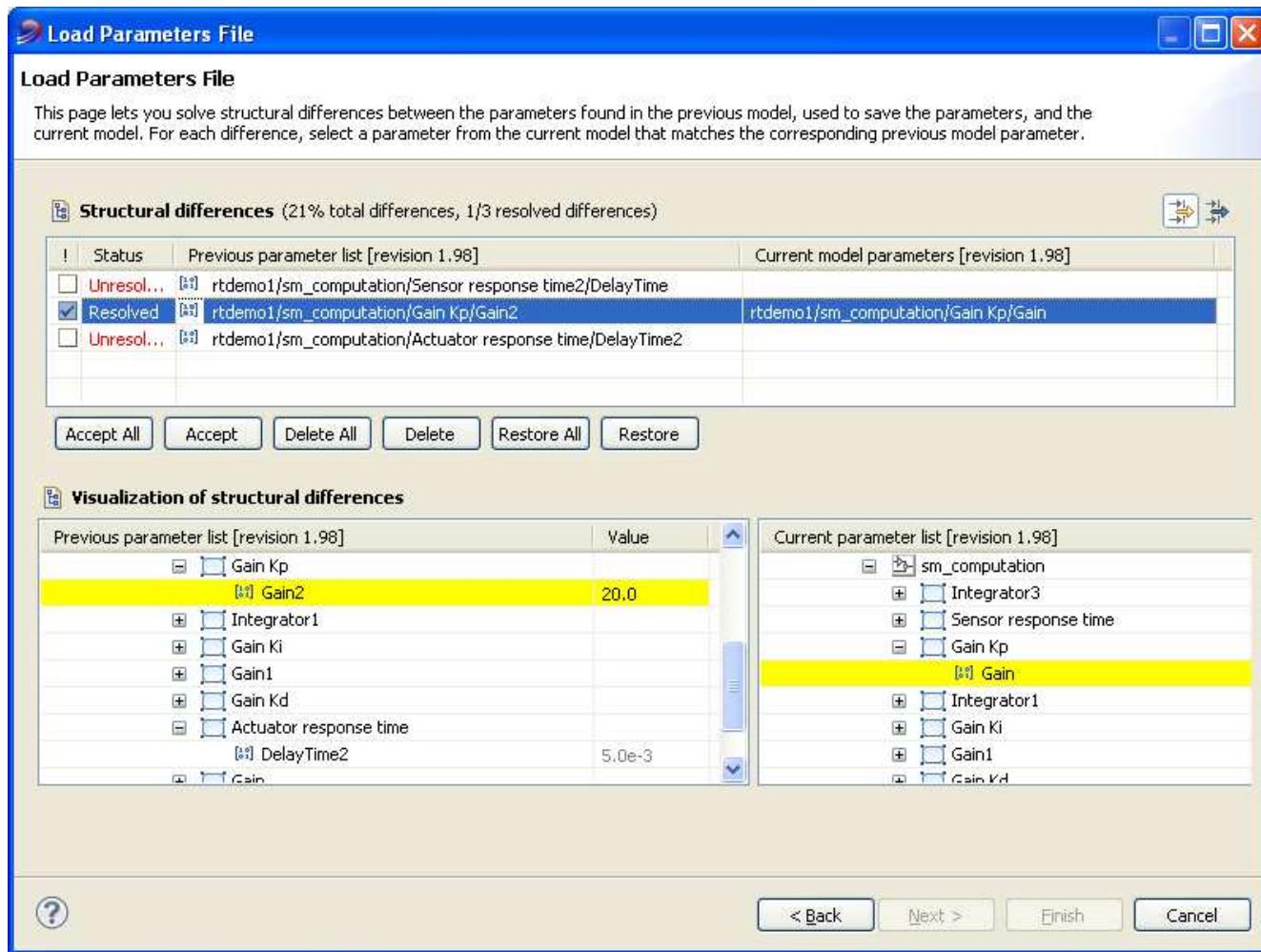
Created on: Parameter file creation date.

Previous Model Information : History information of the model that was used to generate and save the parameter file. Useful when you load a file that was generated using a different model. You may compare this information with the current model information.

Current Model Information : History information of the current selected model in the Project Explorer in which the parameter values will be loaded. Useful when you load a file that was generated using a different model. You may compare this information with the previous model information.

Solving Structural Differences

RT-LAB allows you to load a file that contains structural differences between the previous model content, when the files were saved, and the current model contents. However, you will have to specify a strategy to adopt to solve conflicts for each removed, renamed or moved parameters. The selection of the strategies is performed using the second page.



The **Structural Differences** list, at the top of the page, displays all parameters loaded from the file that do not exist in the current running model because they were deleted, renamed or moved since the writing of the parameter file. Depending on the modifications that were done on the model, you may want to ignore these parameters, or find a matching parameters in the current model.

At the bottom left of the page, you will see the **Previous Parameters** list showing all parameters loaded from the file, organized according to the model's subsystems structure. At the bottom right, you will see the **Current Parameters** list showing all parameters of the current model, organized according to the model's subsystems structure. If there is a mapping between a previous and current model's parameters, it will be highlighted in yellow in each tree. When a parameter is highlighted in both trees, it means that the previous highlighted parameter will be converted to the current highlighted parameter.

When you open the page, all non-existing parameters will have a undesirable status: unresolved or estimated. For each parameter, you will have to accept the estimation, confirm that you want to ignore the parameter, or find a new corresponding matching item in the current model. This process is not automatic but RT-LAB will try to find good estimation for renamed or moved parameters.

Generally, when parameters are deleted from a model, you will ignore them. If they were renamed or moved, they still exist in the current model (with a different path), so you will try to find a corresponding parameter in the current model.

Here are the steps to solve structural differences.

Accepting a estimated parameter

1. Select a parameter from the **Structural Differences** list that is stated as estimated.
2. If you believe the estimation is correct, click the **Accept** button or check the box at the beginning of the line to force RT-LAB to accept the estimation. The value loaded from the file will be applied to the matching parameter of the current model.

Finding a new matching parameter

1. Select any parameter from the **Structural Differences** list.
2. In the **Current Parameter** list, at the bottom right, browse the tree structure and locate the parameter you want to use.
3. Double click the parameter to map this parameter to the previous parameter.
4. The difference status will be resolved. The value loaded from the file will be applied to the matching parameter of the current model.

Ignoring a parameter

1. Select a parameter from the **Structural Differences** list.
2. If the parameter's status is estimated or solved, click the **Delete** button to remove its estimation. Note that you could also right click on the hightligthed parameter in the **Current Parameter** list, at the bottom right of the wizard page, to remove its estimation. The matching item will be cleared and the status will become unresolved.
3. Click the **Accept** button or check the box at the begining of the line to force RT-LAB to ignore the parameter. The value loaded from the file will not be applied to any parameter of the current model.

Restoring parameter

1. Select any parameter from the **Structural Differences** list.
2. Click the **Restore** button to restore the default saved status and estimation of the current parameter.
3. The parameter status will be restored to unresolved or estimated.

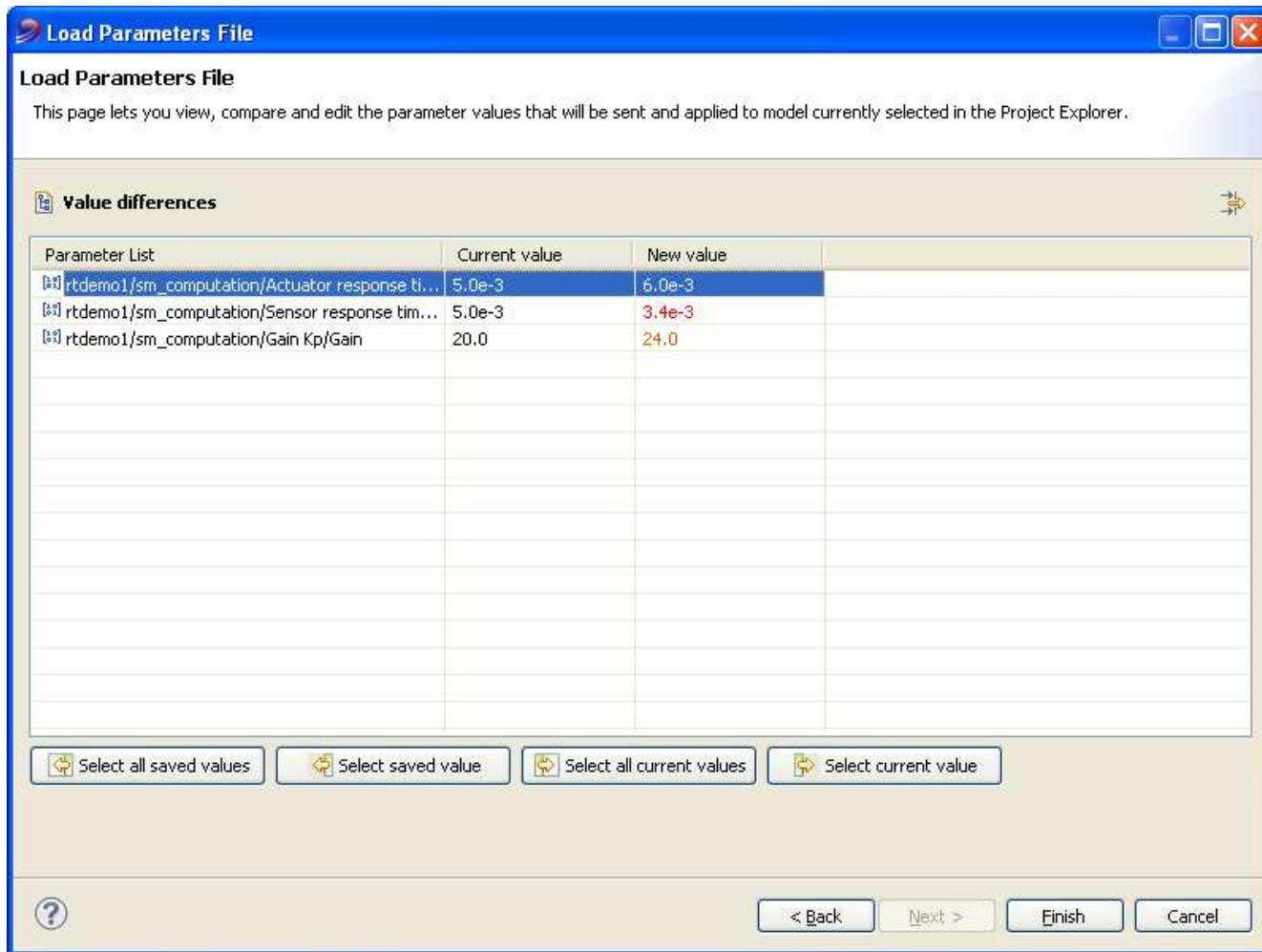
This page also allows you to change the status and estimation of many parameters at the same time by clicking one of the following buttons: accept all, delete all, restore all. The behaviour expected is described above.

This page also allows you to filter items shown in the **Structural Differences** list. By default, only non existing parameters are shown. However, clicking the **show all** button at the top right of the page allows you to view all parameters stored in the file, even those with same unchanged model path. Unchecking the **Show all resolved** button will filter all items that are solved.

When all structural differences are solved, click the **Next** button to view, compare and edit the values loaded from the file.

Viewing, Comparing and Editing Values

The last wizard page allows you to view, compare and edit value loaded from the file.



The **Value Differences** list shows the parameter values that will be sent and applied to the model currently selected in the Project Explorer.

The first column shows the path of parameters in the model diagram. The second column shows the current values of these parameters in the current running model. The last column shows the new values that will be set. This list allows you to compare the values and quickly see what will change when the file is loaded. If modifications are required, you can also edit the values of the third column before applying changes.

If desired, this page also allows you to show all parameters loaded from the file that have the same values as the current model parameter values. To perform this, click the **Show all** button at the top right of the page.

When you are satisfied with the new values, click finish to apply these values to the current model.

Related reference

[Project Explorer](#)

[Save Parameters wizard](#)

Save Parameters wizard

This wizard helps you to save the parameter values of model currently selected in the **Project Explorer**. Here is what this wizard looks like:

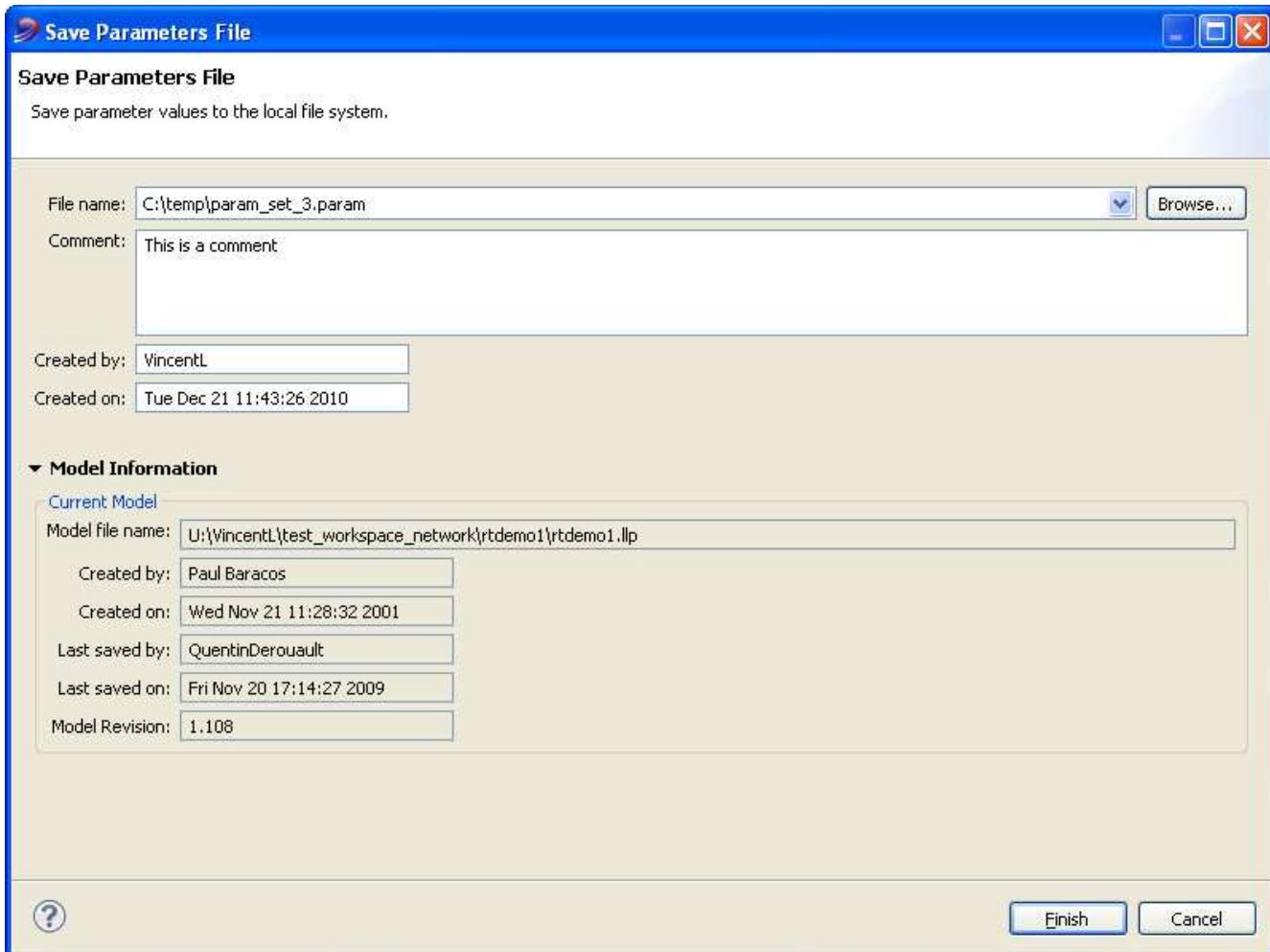


Figure 101:Save parameters dialog

RT-LAB will use the current running model and save all current model values in the specified file. This file may be used later to roll back the parameters. This functionality is particularly useful when tuning parameters.

Description

Filename: File path and name (including the .param extension) where the file will be stored. Use the dropdown arrow to override a file from the history or the browse button to select a file from the file system.

Comment : Comment stored with your parameters.

Created by : File author.

Created on: File creation date.

Model Information : History and revision information of the model that was used to save the parameter file.

Related references

[Project Explorer](#)

[Load Parameters wizard](#)

Dialogs

Contents

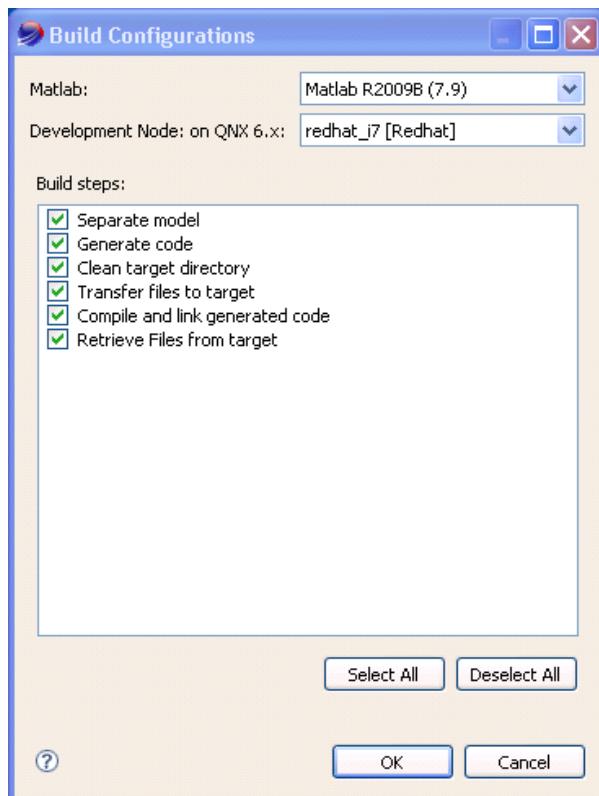
- [Build Configurations](#)
- [RT-LAB Search](#)
- [File Search](#)
- [License Request Dialog](#)

Build Configurations

The **Build configurations** dialog allows to setup the build configuration of your model and then start the build process.

To start the build using the Build configuration dialog select **Simulation > Build** Configurations menu.

Here is what the dialog looks like:



Matlab

The MATLAB version used for during build.

Note :

- This setting is only available when building a Simulink model.
- This setting is available when “**Use same Matlab version than model**” option in Matlab preference page is unchecked.

Development node

The development node used to compile and link the code into an executable. This development node must use the same platform (operating system) as the model. **Note** that this setting is not available when building a model for Window OS.

Build steps

This list displays the list of steps to perform during the build process. Here is a brief description of the steps.

- **Separate model:** When this option is checked, the model is separated into subsystem(s) as specified in the model. This option is always applied to all model.
- **Generate code:** When this option is checked, the C code is generated for the selected model or subsystem(s).

- **Clean target directory:** When this option is checked, the model directory on development target node is cleared. All files will be deleted.
- **Transfer files to target:** When this option is checked, the binary code for the selected model or subsystem(s) and the specified extra-files, if any, are transferred to the development node. This option is not available for Windows platform.
- **Compile and link generated code:** When this option is checked, the generated code is compiled and linked for the selected model or subsystem(s).
- **Retrieve Files from target:** When this option is checked, the binary code for the selected model or subsystem(s) and the specified extra-files, if any, are transferred to the host target. This option is not available for Windows platform.

Select All: Add all steps to the build process.

Deselect All: Remove all steps from the build process.

OK: Applies changes and starts the build process.

Cancel: Closes panel without applying changes.

File Search

This dialog is useful to find files in the workbench. The Search File dialog looks like:

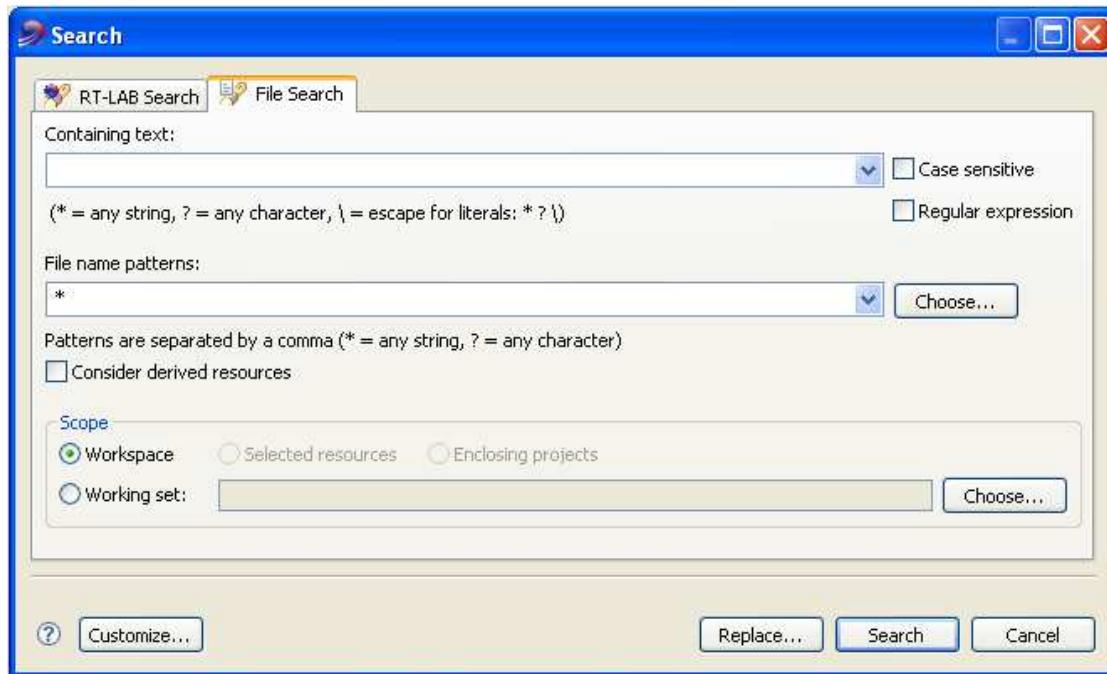


Figure 102:File Search

This is how to search the Workbench for files:

- In the **Containing text** field, type: *it*. The **Containing text** field also displays a list of recently performed searches to select from.
- The **Case sensitive** checkbox can be selected or deselected depending on whether or not a case sensitive search is to be performed. You can also select the **Regular expression** checkbox to enable more powerful searching capabilities. To see what is available in regular expression mode, press Ctrl-Space while the cursor is over the text field to get content assistance that lists the possibilities. For this example, only check the **Case sensitive** box to search for lowercase "it".
- The types of files to include in the search can be specified in the **File name patterns** field. Click **Choose...** to open the Select Types dialog. This dialog provides a quick way to select from a list of registered extensions.
- For the moment, the search will be confined to .txt files. Ensure *.txt is checked and click **OK**.
- In the **Scope** field, specify the files and folders to include in the search. The choices are: Workspace; the currently selected resources in the Workbench; or Working set, which is a named, customized group of files and folders. Leave Workspace, as the scope.
- Click **Search**. At this point, the Search view will be made visible, and it will begin to display the results of the search. The stop button in the Search view can be clicked to cancel the search while it is in progress.

The results of the search are displayed in the [Search View](#).

Related concepts

- [Views](#)
- [Working Sets](#)
- [Toolbars](#)

Related tasks

[Opening views](#)

[Moving and docking views](#)

Related reference

[Search View](#)

RT-LAB Search

This RT-LAB Search dialog allows you to search for RT-LAB elements. To show the tab click **Search > RT-LAB Search**. The results of this search are displayed in the **Search View**.

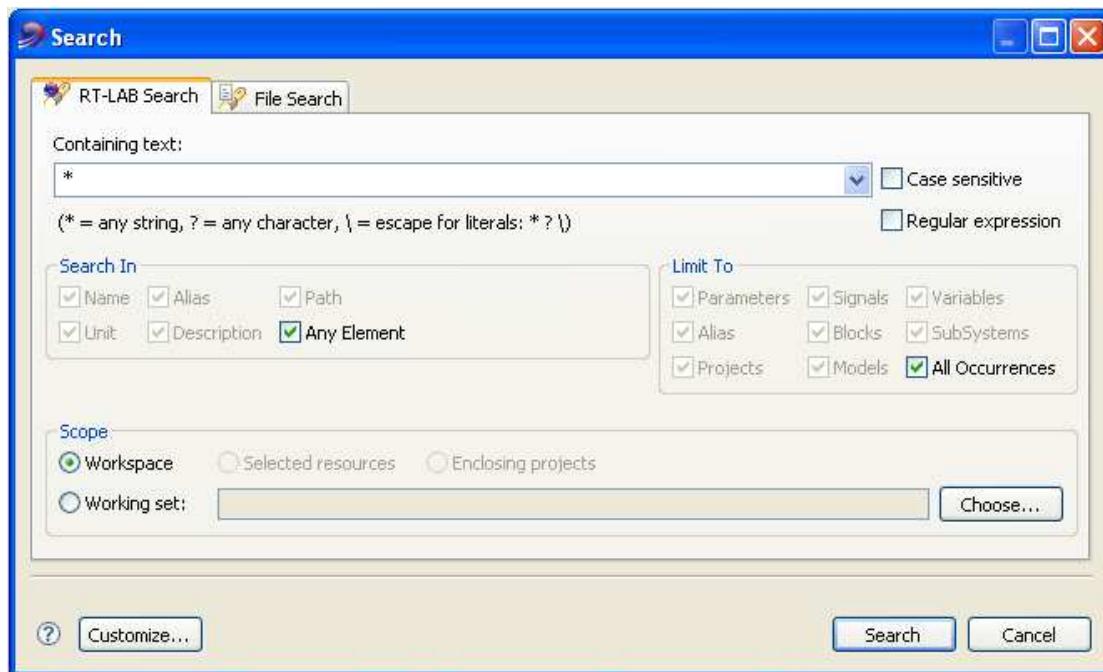


Figure 103:RT-LAB Search Tab

The RT-LAB searching support allows you to quickly find signals, parameters and more RT-LAB objects.

Containing text

In this field, type the expression for which you wish to search, using the wildcard characters listed below the field. This field is initialized based on the current selection.

Depending on what is searched, the search string should describe the element:

- Name: the object name

Examples:

- *Signal1*
- *Error*
- *rtdemo*

- Path: the object path

Examples:

- *rtdemo2/sm_controller/Actuator response time/DelayTime*
- *rtdemo2/sm_controller/Sum3/port1*
- *rtdemo2/sm_controller/Integrator1/InitialCondition*

- Alias: the object alias

Examples:

- *sensor*
- *humidity*
- *level*

- Unit: the unit of a signal or a parameter

Examples:

- *N*

- *km*
- *L*
- Description: the object description
Examples:
 - *Project X*
 - *motor*
 - *machine*
- Any Element: Any element between name, path, alias, unit and description

From the drop-down menu, you can choose to repeat (or modify) a recent search.

Select the **Case sensitive** field to force a case aware search. **Case sensitive** is enabled when a custom search string is entered. Select **Regular Expression** to use the text search patterns for string matching.

Wildcards

The available wildcards for search expressions are displayed in the search dialog:

- "*" matches any set of characters, including the empty string
- "?" matches for any character
- "\\" is the escape for a literal; if you want to search for an asterisk, question mark, or backslash character, type a backslash before it to indicate that you are not using these characters as wildcards (e.g., "*", "\?", or "\\")

Search In

Select where in the scope to search for results:

- Name
- Alias
- Path
- Unit
- Description
- Any Element

Limit To

Select to limit your search results to one of the following kinds of matches:

- Projects
- Models
- SubSystems
- Blocks
- Parameters
- Signals
- Alias
- Variables
- All occurrences

Scope

Select to limit your search results to one of the following scopes:

- **Workspace** - all projects in the workspace are included in this search
- **Selected resources** - only the type hierarchy of the selected element is included in this search
- **Enclosing Projects** - the projects enclosing the currently selected elements
- **Working Set** - only resources that belong to the chosen working set are included in this search

Press **Choose** to select or create a working set.

Related concepts

[Views](#)
[Working Sets](#)
[Toolbars](#)

Related tasks

[Opening views](#)
[Moving and docking views](#)

Related reference

[Search View](#)

License Request Dialog

This dialog is available from the Help menu: **Help > Request a license**. It also appears at startup if your license file is missing or expired.

Here is what the dialog looks like:

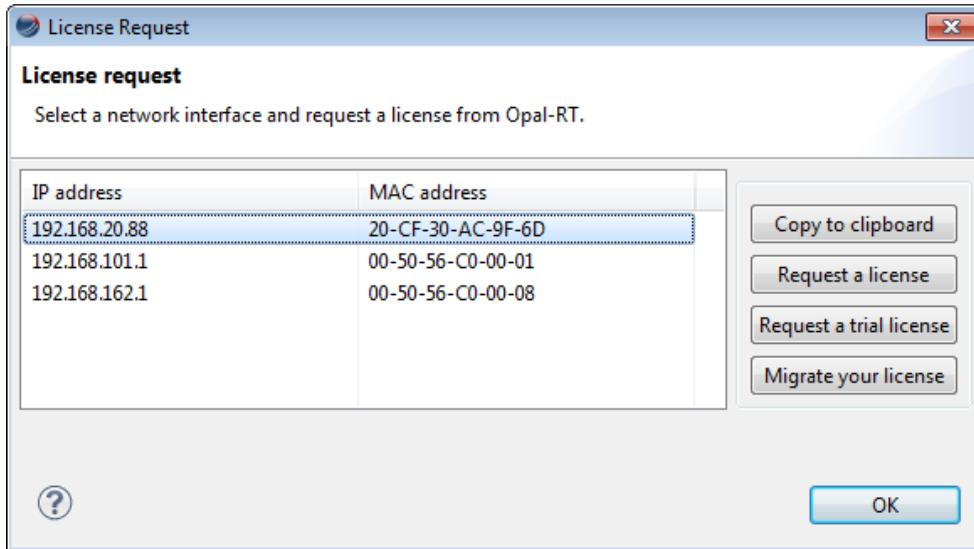


Figure 104: License Request Dialog

The license file will be attached to the selected network interface. The default address is selected when the dialog is opened.

Copy to clipboard: Copy the selected MAC address to the clipboard.

If an Internet connection is available, you can open a pre-filled web form and send the request immediately:

Request a license: Activate a permanent software license.

Request a trial license: Trial Licenses of RT-LAB are available upon request to qualified organizations. A Trial License enables the evaluation of RT-LAB for a 30 days period.

Migrate your license: If you just downloaded an Opal-RT software upgrade or if you are seeking to migrate your Opal-RT software license to a new hardware platform.

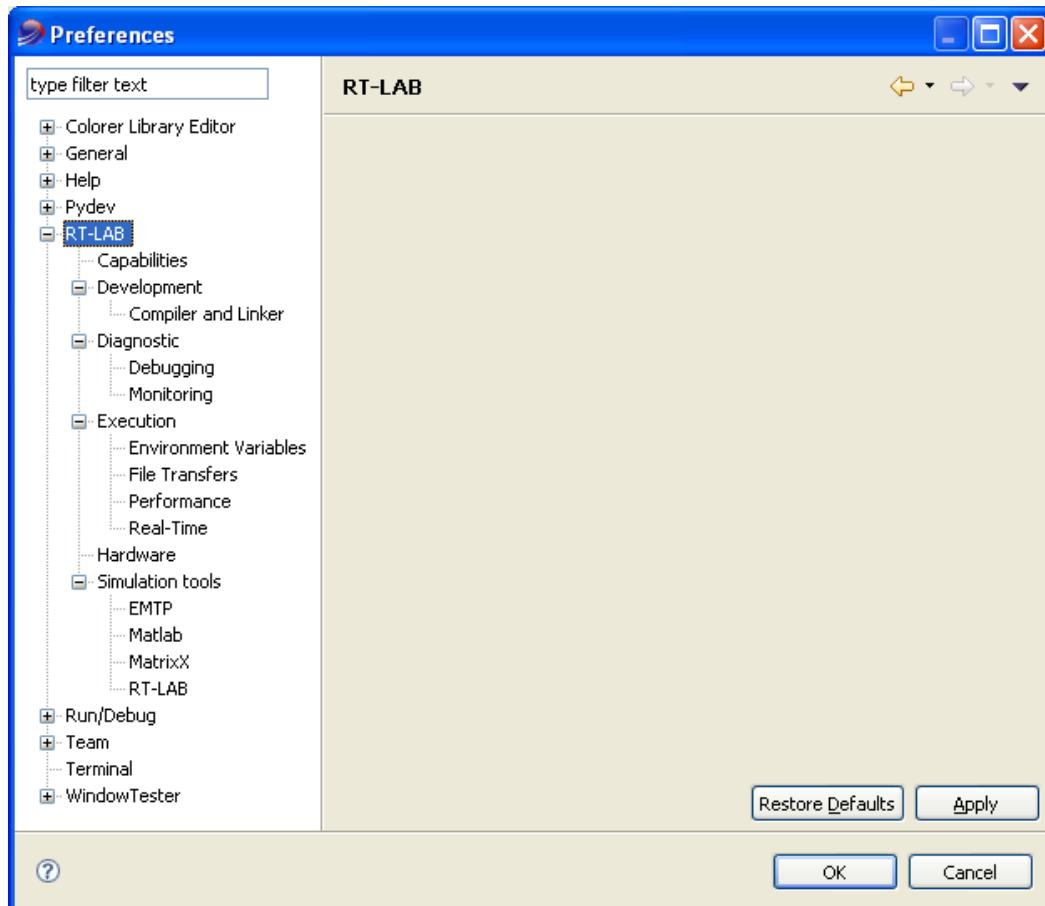
Preferences

Use the **Window > Preferences** dialog pages to set how you want RT-LAB to operate.

You can browse the Preferences dialog pages by looking through all the titles in the left pane or search a smaller set of titles by using the filter field at the top of the left pane. The results returned by the filter will match both Preference page titles and keywords such as "source" and "real-time". However, to find specific functions you may have to search the online help instead.

The arrow controls in the upper-right of the right pane enable you to navigate through previously viewed pages. To return to a page after viewing several pages, click the drop-down arrow to display a list of your recently viewed preference pages.

The preferences dialog displaying the General preferences and RT-LAB preferences:



Preference pages contributed by plug-ins are included in this dialog.

RT-LAB Preferences Pages

Use the **Window > Preferences** dialog page to set the preferences of RT-LAB.

The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

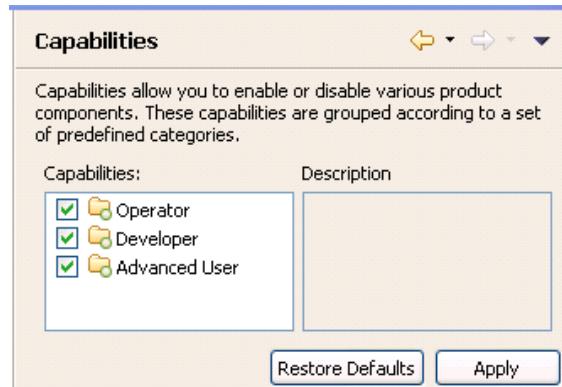
RT-LAB preferences are organized into the following categories:

- [**Capabilities preference page**](#)
- Development
 - [**Compiler and Linker preference page**](#)
- Diagnostic
 - [**Debugging preference page**](#)
 - [**Monitoring preference page**](#)
- Execution
 - [**Environment Variables preference page**](#)
 - [**Files transfers preference page**](#)
 - [**Performance preference page**](#)
 - [**Real-time preference page**](#)
- [**Hardware preference page**](#)
- [**Simulation Tools preference page**](#)
 - [**EMTP preference page**](#)
 - [**Matlab preference page**](#)
 - [**RT-LAB preference page**](#)
- [**Target Detection preference page**](#)

Capabilities preference page

Use the **Window > Preferences > RT-LAB > Capabilities** dialog page to set the preferences concerning the RT-LAB capabilities.

Here is what the Capabilities preference page looks like:



The capabilities preference page allows you to enable or disable various product components of RT-LAB such as Operator or Developer environments. By default, no capabilities are active and only the basic functionalities of RT-LAB are active.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Operator: This capability enables all features that allow user to manage consoles and make some changes to the parameters and aliases of the models. It enables all components making data acquisition and monitoring and also basic operations on the simulator as loading, executing, pausing or resetting the models. (default value=false)

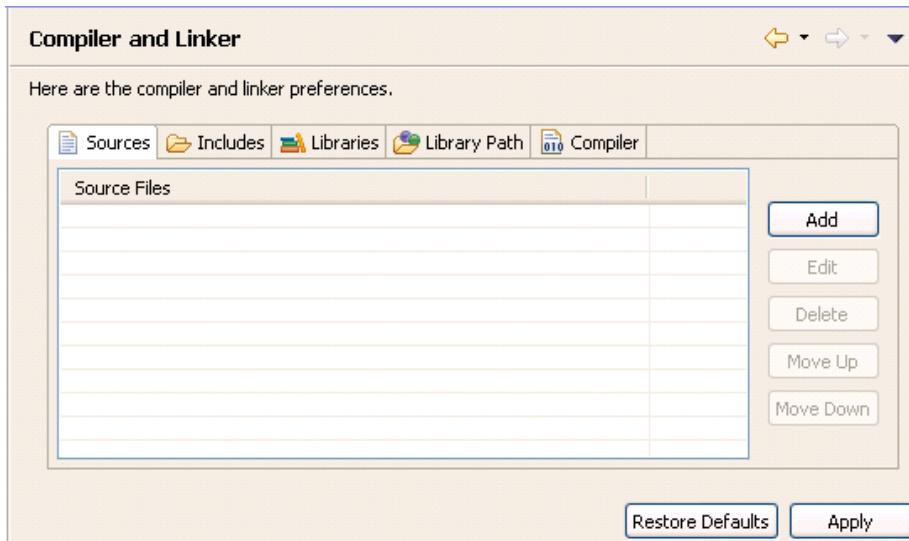
Developer: This capabilities enables all features related to the development of models, scripts, and codes. It also enables all features of the operator capabilities. (default value=false)

Advanced User: This capabilities enables all products components including operator and developer capabilities. Moreover, it also enables other unclassified features related to Eclipse and external plugins. (default value=false)

Compiler and Linker preference page

Use the **Window > Preferences > RT-LAB > Development > Compiler and Linker** dialog page to set the preferences concerning the RT-LAB compiler and Linker.

Here is what the Compiler and Linker preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Sources: Names of source files to be compiled in addition to the code generated by RTW. Used to incorporate user-written code into the model executable (e.g. file1.c file2.c file3.c)

Includes: Target path(s) where user include files can be found. Used to search for include files required to compile user-written code, when the include files do not reside in already-searched paths.

Libraries: Names of user-specified libraries to be used when incorporating user-written code, in addition to the standard system libraries (Matlab and RT-LAB).

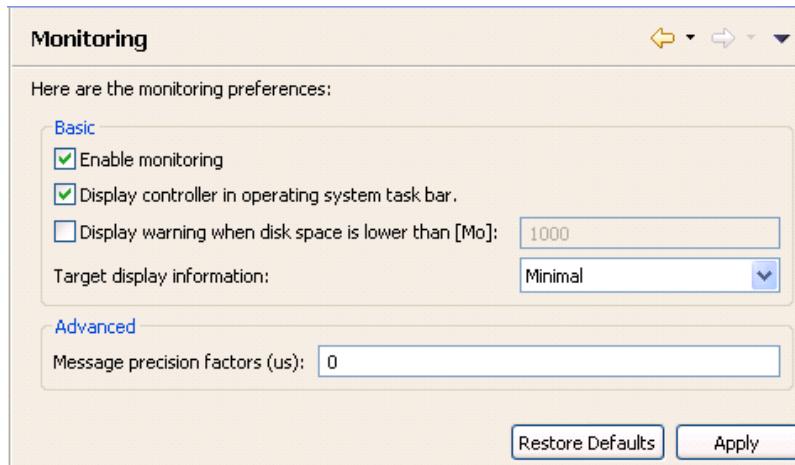
Library path: Target path(s) where user libraries can be found. Used to search for libraries other than the MATLAB and RT-LAB libraries when incorporating user-written code.

Compiler command: User command to be called on the target during the compilation process.

Monitoring preference page

Use the **Window > Preferences > RT-LAB > Diagnostic > Monitoring** dialog page to set the preferences parameters concerning the RT-LAB monitoring.

Here is what the Monitoring preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Enable monitoring: Enables or disables the monitoring during the execution of the model. (default value=true)

Display controller in operating system task bar: Displays RT-LAB Controller application in the taskbar. Unchecked by default. (default value=false)

Display warning when disk space is lower than [Mo]: When checked, a warning message is displayed when the disk space is lower than the value specified. (default value=false)

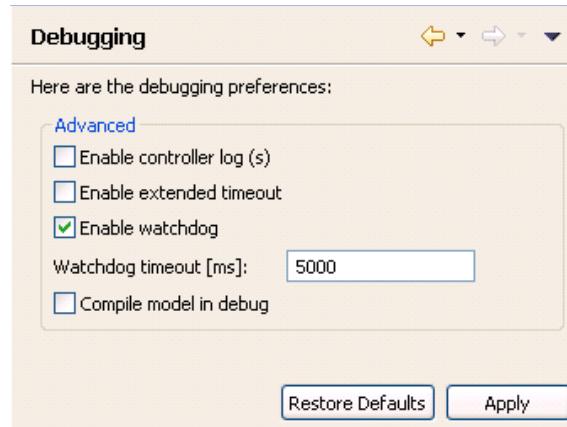
Target display information: Represents the amount of information that is displayed in the display window during load and execution. Setting the value from MINIMAL to EXHAUSTIVE allows the user to retrieve more information about communication and I/Os initialization for example. This can be used to debug unexpected model behaviors. (default value="minimal")

Message precision factor (us): Defines the precision of the timer reporting the time required to perform one calculation step in microseconds. The default value is 0 (no printout). Value=1 prints the step size of the model every 1,000,000 steps with a precision of 1 µs. Value=100 prints the model's step size every 10,000 steps with a precision of 100 µs and so on. (default value=0)

Debugging preference page

Use the **Window > Preferences > RT-LAB > Diagnostic > Debugging** dialog page to set the preferences concerning the debugging of RT-LAB model when running on the target nodes.

Here is what the debugging preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Enable controller log (s): Creates a file called "Controller.log" in the RTLAB_ROOT\common\bin folder for debugging use. This file reports all operations perform on the model during the life-cycle of your project. Note also that it reports also other internal operation related to the controller of your project. Unchecked by default. (default value=false)

Enable extended timeout: All timeouts are extended when this option is enabled. It is typically used when debugging the model. (default value=false)

Enable watchdog: Enables or disables the Watchdog functionality. The Watchdog ensures that all nodes are running normally. There are as many watchdogs as there are computation subsystems in the model. At regular user-defined time intervals, the watchdog verifies that the model is still running and that processes are still active. If an error occurs, the watchdog stops the simulation. This option is not available on Windows target.(default value=true)

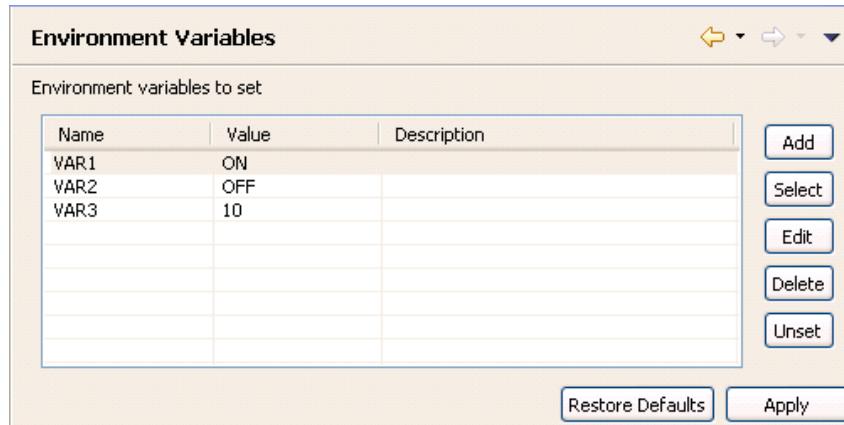
Watchdog timeout [ms]: The regular user-defined time interval the watchdog verifies the model is still running. 5000 ms by default. (default value=5000)

Compile model in debug: When this option is enabled, the compiler's optimization options are removed and replaced by debug options during model compilation. This option is useful when debugging a model. (default value=false)

Environment Variables preference page

Use the **Window > Preferences > RT-LAB > Execution > Environment Variables** dialog page to add environment variables.

Here is what the Environment Variables preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Name: Name of the user variable.

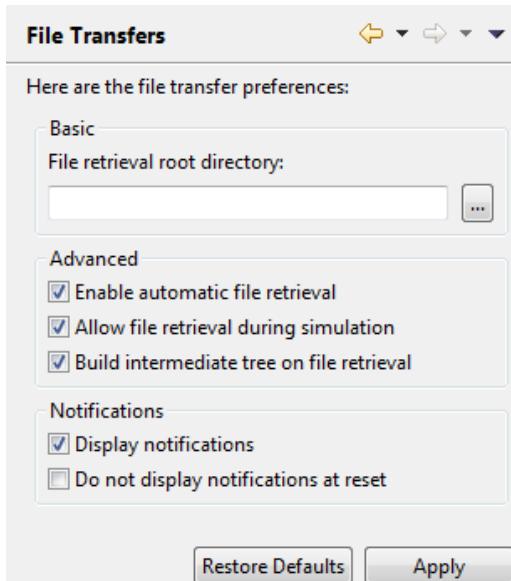
Value: Value of the user variable.

Description: Description of the user variable.

Files transfers preference page

Use the **Window > Preferences > RT-LAB > Execution > File transfers** dialog page to add some files to be transferred.

Here is what the File Transfers preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

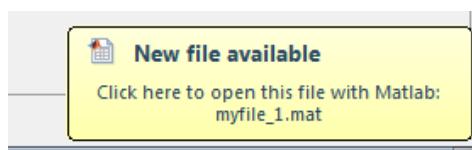
File retrieval root directory: Directory where the files retrieved by RT-LAB after the model execution will be copied.

Enable automatic file retrieval: When this option is enabled, RT-LAB retrieves files generated by the model simulation on the target nodes and transfers them to the host computer. (default value=true)

Enable file retrieval during simulation: If this option is enabled, RT-LAB retrieves the files generated by the model simulation as soon as they are created on the target. If this option is disabled, RT-LAB will retrieve those files after the model resets.

Build intermediate tree on file retrieval: Specifies if intermediate folders are generated or not in the retrieve directory. (default value=true)

Display notifications: Displays a popup notification each time a file has been retrieved:



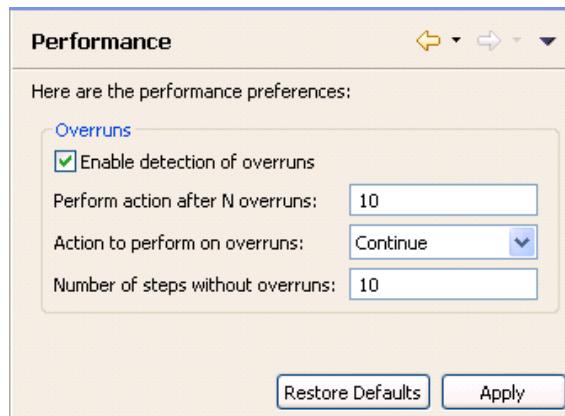
Clicking on the notification opens the file with an appropriate editor or with Matlab, depending on the file extension.

Do not display notifications at reset: Prevents RT-LAB from displaying notifications when files are retrieved after a model has been reset. This is mostly useful when many files are retrieved, since there would be one notification per file.

Performance preference page

Use the **Window > Preferences > RT-LAB > Execution > Performance** dialog page to set the performance preferences.

Here is what the Performance preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Enable detection of overruns: (Only applies to models ran in hardware/software synchronized modes.) When this parameter is not checked, RT-LAB does not detect overruns. When this parameters is checked, RT-LAB detects overruns and performs the action specified in the "Action to perform on overruns" field. This property has no effect in simulation mode. (default value=true)

Perform action after N overruns: Performs the action specified in the "Action to perform on overruns" field when the number of overruns detected by RT-LAB reaches this number. (Only applies when "Action to perform on overruns" is set to Reset or Pause.) (default value=10)

Action to perform on overruns: Type of action to perform when overruns are detected. (default value="Continue")

Here are the available types:

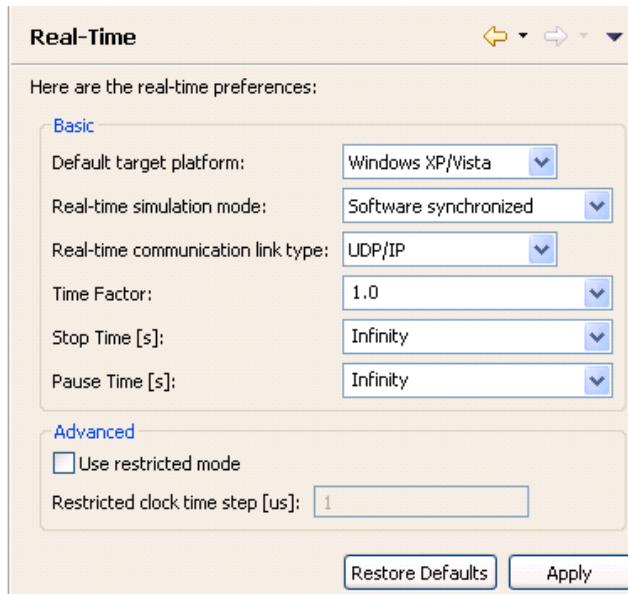
- Continue: No action is perform ; Number of overruns will simply be displayed during the next pause/reset of model.
- Reset: A reset of the model is performed when the number of overruns reaches the value of the "Perform action after N overruns" field.
- Pause: A pause of the model is executed when the number of overruns reaches the value of the "Perform action after N overruns" field.

Number of step without overruns: Prevents RT-LAB to detect overruns during the first steps of simulation. (default value=10)

Real-time preference page

Use the **Window > Preferences > RT-LAB > Execution > Real-Time** dialog page to set the real-time preferences.

Here is what the Real-Time preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Default target Platform: Allows the user to choose the target platform on which to run the simulation (QNX6, Redhat, Redhawk, Windows XP/Vista/7 system). (default value="QNX 6.x")

Real-time simulation mode: Enables the user to define how the model's simulation can be executed on a QNX6, RedHat, or Windows XP/Vista/7 system. (default value="Software synchronized")

- Simulation: in this mode, the model is not synchronized; the model starts a new computation step as soon as the previous one is completed (the model runs as fast as possible).
- Simulation with low priority: this simulation mode can be used to run a simulation in the background while working with other applications on the same computer.

Note: Please note that the Simulation with low priority mode is only available when working with a Windows target.

- Software Synchronized: in this mode, real-time synchronization of the entire simulation is achieved by the OS, using the CPU clock as a reference. Depending on the resolution available on the OS, some sampling times may not be obtained using this mode (e.g. on QNX6.1 the smallest sampling time available is 500 µs).

Hardware Synchronized: in this mode, an I/O board clock is used to synchronize the entire simulation.

Real-time communication link type: Sets the target cluster's communication medium. (default value="UDP/IP")

Here are the available choices for each target platform.

- XP/Vista/7 system: UDP/IP.
- QNX 6.x: OHCI, UDP/IP.

RedHat: OHCI, UDP/IP, SCI, INFINIBAND.

Time Factor: This value, when multiplied by the model's basic calculation step (or fixed step size), supplies the final calculation step for the system. The time factor can only be changed when the simulation execution is paused and is only available in Synchronized (hardware/software) mode. (default value=1.0)

Note: This parameter enables you to change the I/O acquisition speed. Because the model always uses the basic calculation step, changing the time factor for the I/O may give results which differ from the offline simulation. This parameter should be used only to determine the simulation's minimum calculation step. Once this step is determined, this parameter must be brought back to a value of 1. The model's basic calculation step must be updated accordingly by setting the correct value in the Simulink model and by recompiling it.

Stop time: This value (in seconds) specifies when the model will stop. If value is "Infinity", the model will execute forever. You don't need to recompile the model to apply a new value. (default value="Infinity")

Pause time: This value (in seconds) specifies when the model will pause. If value is "Infinity", the model will execute until a reset is done except if a stop time is specified. (default value="Infinity")

Use Restricted mode: (For advanced user only): This setting can be used only when the model's current target platform is QNX 6.x. When this platform is selected, a given model can be loaded either in 'Free Clock' mode or in 'Restricted Clock' mode. (default value=false)

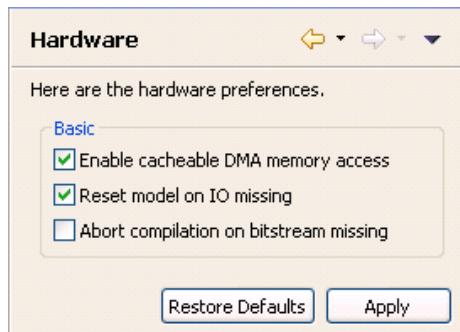
- In Free-Clock mode, the user can load one non-Software Synchronized model on a given target node and is free to use any time-step value (within the boundaries of the QNX 6.x restrictions). If the user tries to load another non-Software Synchronized model on the same target node, it will be stopped with a relevant error message.
- In Restricted-Clock mode, the user can load a series of non-Software Synchronized models on a given node. In this mode, each model time-step must be consistent (i.e. a multiple of) with a specific base Clock Period that should be specified by the user. Each time an additional model is being loaded on this specific target, its time-step will be checked against the current base Clock Period to validate the load operation.

Restricted clock time step [us]: (For advanced user only): Defines the Clock Period in microseconds specified by the user for the Use Restricted mode option. (Only available when the Use Restricted mode option is enabled).

Hardware preference page

Use the **Window > Preferences > RT-LAB > Hardware** dialog page to set the hardware preferences.

Here is what the File Hardware preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Enable cacheable DMA memory access: When this option is checked, RT-LAB will enable the cache of the memory areas used by DMA buffers yielding to faster transfer rates and increased model performances. However, this option should be used with caution since it can affect proper reading of I/O data using DMA buffers. This option is available on QNX6 only. (default value=true)

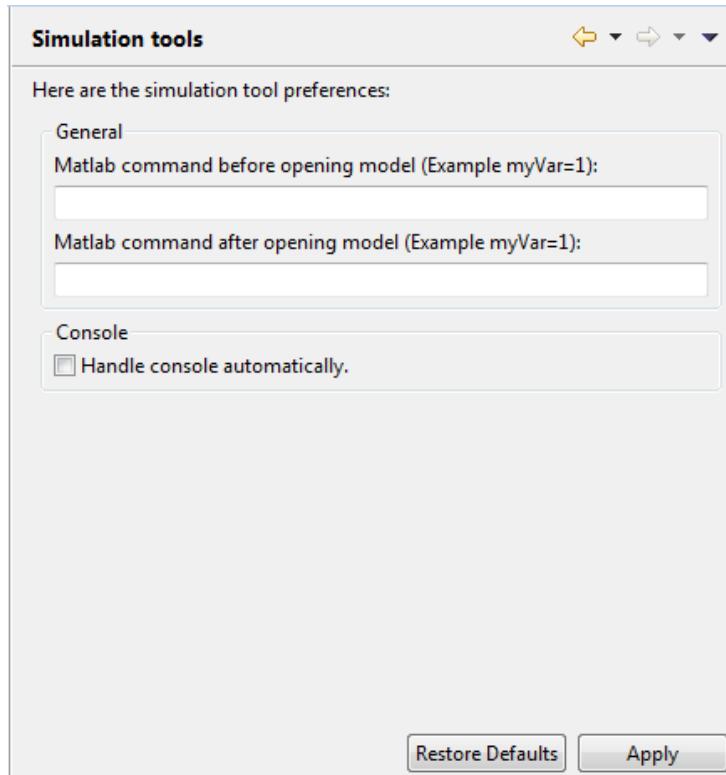
Reset model on I/O board missing: When this option is checked and the model is loaded, RT-LAB will stop loading and will generate an error, if some I/O boards required by the model are not detected on the target(s). RT-LAB will generate only a warning if this setting is unchecked. Note that only a few I/O boards (mainly the Opal-RT TestDrive boards) support this option. Most I/O blocks force a reset of the model if the boards are not detected at load time. (default value=true)

Abort compilation on bitstream missing: When this option is enabled, if some bitstreams required by a model are not present in the model directory, RT-LAB will stop the compilation and generate an error. RT-LAB will generate only a warning if this setting is disabled. (default value=false)

Simulation Tools preference page

Use the **Window > Preferences > RT-LAB > Simulation Tools** dialog page to set the simulation tools preferences.

Here is what the Simulation Tools preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

Matlab command before opening model: Matlab/Xmath command to execute before opening the model. For example, MyInitFile.m, where MyInitFile includes commands that initialize model variables. (default value="")

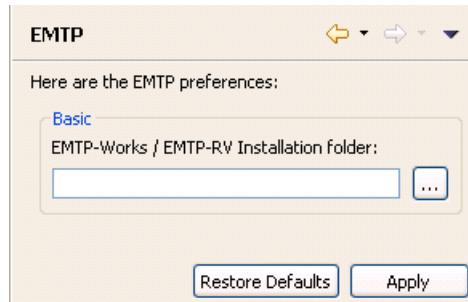
Matlab command after opening model: Matlab/Xmath command to execute after opening the model. For example, MyInitFile.m, where MyInitFile includes commands that initialize model variables. (default value="")

Handle console automatically: Enables the console to be automatically opened and closed, when a model is Loaded or Reseted. The console starts automatically at the first model's execution. (default value=false)

EMTP preference page

Use the **Window > Preferences > RT-LAB > Simulation Tools > EMTP** dialog page to set the EMTP preferences.

Here is what the EMTP preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

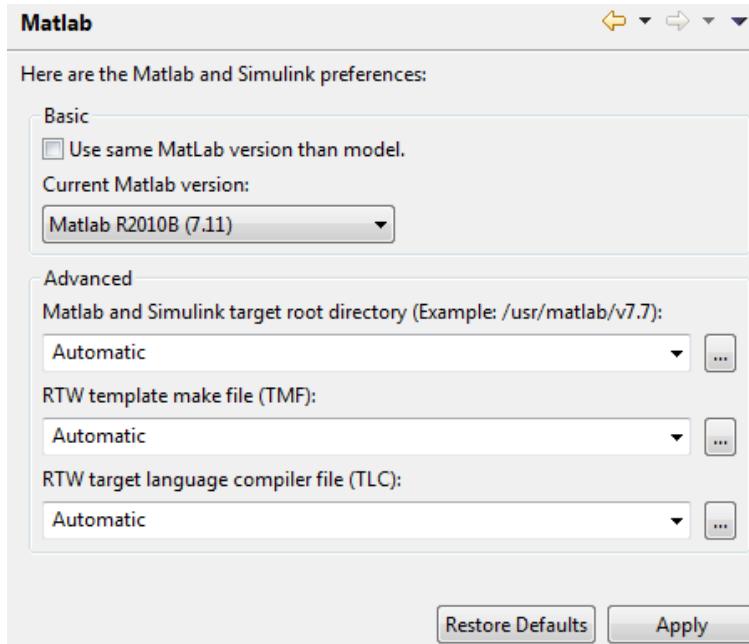
The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

EMTP-Works /EMTP-RV Installation folder: Path where the EMPT tool is installed.

Matlab preference page

Use the **Window > Preferences > RT-LAB > Simulation Tools > Matlab** dialog page to set the matlab preferences.

Here is what the Matlab preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Use same Matlab version than model: Use same Matlab version than model to open and compile Simulink mode. If Matlab version of model is not supported or not installed, use Current Matlab version. (default value="unchecked")

Note : if, in opened project, model is saved in new Matlab version, the new Matlab version will be take account next time your project will be opened.

Current Matlab version: Matlab version used to open and compile Simulink mode. Note: this setting will be changed if the user opens the model by choosing "Edit with" and a new matlab version. (default value="Matlab R2011B")

Matlab and Simulink target root directory: Path to be used instead of /usr/matlab to find the MATLAB include and source files. RT-LAB supports multiple versions of Matlab by keeping copies of previous versions in paths other than /usr/matlab. Automatic setting means the target path is automatically set based on the MATLAB version used on the Command Station. (default value="automatic")

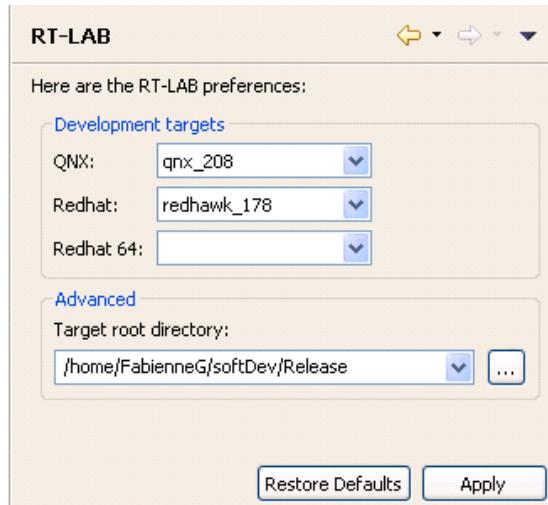
RTW template make file (TMF): Specifies which template makefile to use during the compilation. When Automatic is selected, the template makefile corresponding to MATLAB version is automatically selected. This option is available for SIMULINK models only. (default value="automatic")

RTW target language compiler file (TLC): Specifies which tlc file to use during the compilation. When Automatic is selected, the tlc file corresponding to MATLAB version is automatically selected. This option is available for SIMULINK models only. (default value="automatic")

RT-LAB preference page

Use the Window > Preferences > RT-LAB > Simulation Tools > RT-LAB dialog page to set the RT-LAB preferences.

Here is what the RT-LAB preference page looks like:



The value of these parameters are used when a new model is created. If you change a property from Preference page, each new model will inherit from Properties defined into Preference page but the existing models will stay unchanged.

The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

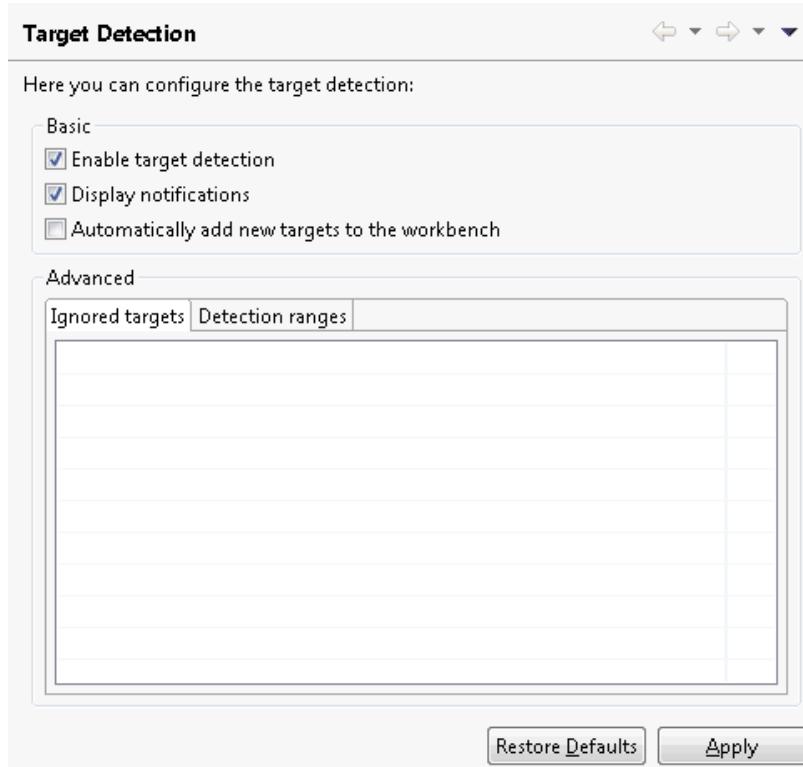
Development targets QNX/Redhat/Redhat64: Identifies the target node where the compilation is done. Target nodes must be defined in the workbench, by using the New Target wizard, before the development node can be chosen. For more details, see New Target Wizard.

Target root directory: Path to be used to find the RT-LAB executables, include files and libraries or the target nodes. This allows previous versions of RT-LAB to be used side-by-side with newer versions. The set-up script makes the last version installed the default system but, this option can be used to specify a version based on its path. Automatic setting means the target path is automatically set based on the last installed RT-LAB version. (default value="Automatic")

Target Detection preference page

Use the **Window > Preferences > RT-LAB > Execution > Target Detection** dialog page to configure the RT-LAB target detection.

Here is what the Target Detection preference page looks like:



The "Restore Defaults" button enables the user to restore the defaults value of the parameters.

Enable target detection: Uncheck this option to completely disable automatic target detection. Target detection can still be manually done by right-clicking on the "Targets" item of the **Project Explorer** and selecting the "Discover targets" menu.

Display notifications: Displays a popup notification each time a target is detected:

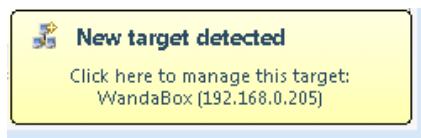


Figure 105:"New target" notification

Clicking on the notification opens the **Detected Targets Wizard**.

Automatically add new targets to the workbench: New targets are added to the workbench as soon as they are detected.

Ignored targets: If you do not want to be notified when a particular target is connected to the network, simply add its IP address to this table. This table is also automatically populated by the [Detected Targets Wizard](#).

Detection ranges: Here you can limit the target detection to some IP addresses by defining ranges. Any target outside these ranges will not be detected by RT-LAB.

Note: Ignored targets and targets outside the detection range can still be used in the Workbench, but you will have to add them manually by using the [New Target wizard](#).

General preferences pages

Use the **Window > Preferences** dialog page to set the general preferences of the RT-LAB workbench

Accessibility preference page

You can change the following preferences on the General > Editors > Text Editors > Accessibility preference page:

Option Description Default

Use custom caret Replaces the original caret (the marker that indicates where the next character will appear) with a custom caret and shows a different caret for Overwrite and Insert modes. On

Enable thick caret

Replaces the original caret with a more visible, thicker caret. On

Use characters to show changes on line number bar

Quick Diff shows the changes in a vertical ruler using colors. Color-blind persons can enable this option to show differences with different characters in the line number ruler. Off

Annotations preference page

The following preferences can be changed on the General > Editors > Text Editors > Annotations preference page.

Show in Text as

This option controls whether the selected annotation type is shown in the text. The corresponding text will be underlined with squiggles or highlighted.

Show in Overview ruler

This option controls whether the overview ruler on the right side of the text editor is shown.

Show in Vertical ruler

This option controls whether the selected annotation type is shown in the vertical ruler.

Color

This option controls the color for the selected annotation type.

Capabilities preference page

The capabilities preference page allows you to enable or disable various product components of RT-LAB such as Operator or Developer environments. By default, no capabilities are active and only the basic functionalities of RT-LAB are active.

Note: Some capability selections have dependencies on other capabilities, disabling a required capability while leaving dependant capabilities enabled will only result in them becoming re-enabled.

RT-LAB: This capability enables all features related to RT-LAB. **Do not disable this capability.**

RT-LAB Operator: This capability enables all features that allow user to manage consoles and make some changes to the parameters and aliases of the models. It enables all components making data acquisition and monitoring and also basic operations on the simulator as loading, executing, pausing or resetting the models. (default value=false)

RT-LAB Developer: This capabilities enables all features related to the development of models, scripts, and codes. It also enables all features of the operator capabilities. (default value=false)

RT-LAB Advanced User: This capabilities enables all products components including operator and developer capabilities. Moreover, it also enables other unclassified features related to Eclipse and external plugins. (default value=false)

Development: Use the Eclipse SDK to develop applications.

Team: Use configuration management systems to manage resources.

Appearance preference page

You can change the following preferences on the General > Appearance preferences page.

Current

presentation Specify the currently active presentation (look and feel).

Override presentation settings

Locally override the settings from the current presentation's defaults.

Editor tab positions

Specify either top or bottom to indicate where you want tabs for stacked editors to appear. Top

View tab positions

Specify either top or bottom to indicate where you want tabs for stacked views to appear. Top

Perspective switcher positions

Specify the location of the perspective switcher bar. Top right

Current theme

Specify the currently active theme (color and font set). Default (current)

Show text on perspective bar

Specify whether labels should be shown in the perspective bar as well as icons. Enabled

Show traditional style tabs

Specify whether traditional (square) tabs should be used in place of the curved tabs. Disabled

Enable animations

Enable/disable the feature where views animate to their location when closed or opened. Enabled

Colors and Fonts preference page

Many of the fonts and colors used by RT-LAB components can be set using the **General > Appearance > Colors and Fonts** preference page.

A tree is used to navigate among and show a short preview of the various colors and fonts. The current face (but not size) of any font is previewed in its label. Colors are previewed in the icon associated with its label. Additionally, some categories (Workbench in particular) provide a more detailed preview of their contributions. This preview is shown below the description area if available.

Font settings can be changed either by selecting the font from the list and clicking **Use System Font** to choose the Operating System font setting or by clicking **Change** to open up a font selection dialog. **Reset** can be used to return to the default value.

Color settings can be changed by clicking **color** to the right of the tree area when a color is selected. **Reset** can be used to return to the default value.

The Colors and Fonts text field can be used to filter the contents. Simply type in an entry and any matching results will remain in the tree view.

Descriptions and previews are provided when the Workbench colors and font settings are selected.

Compare/Patch preference page

The following preferences can be changed on the **General > Compare/Patch** page.

General options

Open structure compare automatically

This option controls whether a structure compare is automatically performed whenever a content compare is done. Turn this option off if you don't want to see the structural differences.

Show structure compare in Outline view when possible

If this option is on, structure compare will be displayed in the Outline view whenever it is possible.

Show additional compare information in the status line

If this option is on, additional information about a change is shown in the status line. Turn this option on if you are interested in additional information about a change.

Ignore white space

This option controls whether or not whitespace changes are shown in the compare viewer. Turn this option on if you want to see changes in whitespace.

Automatically save dirty editors before browsing patches

This option controls whether any unsaved changes are automatically saved before a patch is applied. Turn this option on if you want to save changes automatically.

Filtered Members

This option allows you to filter members that should be excluded from 'Compare With Each Other'.

Note: The names in the list must be separated by a comma.

Text Compare options

Synchronize scrolling between panes in compare viewers

The two comparison viewers will "lock scroll" along with one another in order to keep identical and corresponding portions of the code in each pane side-by-side. Turn this option off if you do not want the compare viewers to lock scroll.

Initially show ancestor pane

Sometimes you want to compare two versions of a resource with the previous version from which they were both derived. This is called their common ancestor, and it appears in its own comparison pane during a three way compare. Turn this option on if you want the ancestor pane to always appear at the start of a comparison.

Show pseudo conflicts

Displays pseudo conflicts, which occur when two developers make the same change (for example, both add or remove the exact same line of code or comment). Turn this option on if you want pseudo conflicts to appear in compare browsers.

Connect ranges with single line

Controls whether differing ranges are visually connected by a single line or a range delimited by two lines.

Highlight individual changes

Controls whether the individual changes inside conflicts are highlighted.

When the end/beginning is reached while navigating an element

Use this option to configure what occurs when the end/beginning is reached while navigating an element.

-
- Prompt: If this option is on and you selected to compare a single element you will be asked whether you want to go to the beginning/end of the element after the end/beginning is reached. If you are comparing two or more elements you will be asked whether you want to go to the beginning/end of the current element or to go to the next/previous element. Moreover, if you choose to remember your decision, this option will be changed to one of the below respectively.
 - Loop back to the beginning/end: When this option is on, the selection will move back to the beginning/end after you reach the end/beginning of an element.
 - Go to the next/previous element: If you are comparing two or more elements and this option is on after you reach the end/beginning of an element the next/previous element will be opened.

Content Types preference page

The **General > Content Types** preference page enables you to edit content types and their associated file names and character sets. You can also associate arbitrary file names or file extensions with content types. A content type acts as a description of a certain class of files (for instance, XML files). RT-LAB uses this description in various scenarios, such as editor look-ups and file comparisons.

To access the **Content Types** preference page, select **Window > Preferences > General > Content Types**.

By selecting a content type in the topmost tree, you can alter the file names and extensions that are associated with it.

Note: Certain items will be marked as "locked". An item is locked if it is one of the associations provided by the plug-in that declares the content type. In other words, you can remove only user-contributed associations.

Adding an association is as simple as clicking **Add....** A dialog prompts you to enter the file name or extension.

In addition to adding and removing file names or extensions, you can also set the default encoding for a given content type. To do this, simply enter the encoding name in the provided field and click

Update.

Editors preference page

You can change the following preferences on the **General > Editors** preference page:

Size of recently opened files list

Each file that is opened in an editor is stored in a list of recently used files in the File menu. This option controls the number of files that is displayed in that list.

Show multiple editor tabs

Specifies whether you wish to show multiple editor tabs. If off, editor workbooks have one large tab and all non-visible editors are accessible only from the chevron.

Close editors automatically

Specifies whether or not to re-use editors in the Workbench. If on, you may specify the number of editors to use before they are recycled (the default is 8). You can also specify if a prompt dialog should be opened or if a new editor should be opened when all editors are "dirty" (have unsaved changes). Once it is turned on, the Pin Editor action is added to the toolbar and editor tab menu. Pinned editors are not recycled.

File Associations preference page

On the **General > Editors > File Associations** preference page, you can add or remove file types recognized by the Workbench. You can also associate editors with file types in the file types list.

File types list

Add...

Adds a new file or file type (extension) to the predefined list. In the resulting New File Type dialog, type the name of a file or a file extension. If you are adding a file extension, you must type either a dot or a "*" before the file type (e.g., ".xml" or "*.*xml" as opposed to simply "xml").

Remove

Removes the selected file type from the list

Associated editors list

Add...

Adds a new editor to the list of editors associated with the file type selected above. In the resulting Editor Selection dialog, you can choose an editor to launch either inside the Workbench (internal) or outside the Workbench (external); click Browse to locate an editor yourself if the editor you want is not displayed in the list.

Remove

Removes the association between an editor and the file type selected above. Note: Any editor that is bound by content type may not be removed from this list. Currently, there is no mechanism available to remove these editors.

Default

Sets the selected editor as the default editor for the file type selected above. The editor moves to the top of the Associated Editors list to indicate that it is the default editor for that file type.

Help preferences

On the **Help** preferences page, you can indicate how to display help information.

Use external browsers

If embedded web browser is supported on your system, help window uses an embedded help browser to display help contents, whenever possible, and this option is available. Select it, to force help to use external browsers. Use "Web Browser" preference page to select browser to use.

Open window context help

This option allows you to determine whether the window context help will be opened in a dynamic help view or in an infopop.

Open dialog context help

This option allows you to determine whether the dialog context help will be opened in a dynamic help section of help view or in an infopop.

Open help view documents

This option allows you to determine whether the documents selected in the help view will be opened in-place or in the editor area.

Note: Selection performed on this page can affect how the help view is presented. If the selected browser is not fully compatible with Internet Explorer or Mozilla, or has JavaScript disabled, the help view shown in the browser might be a simplified version.

General preference page

General settings for the Workbench. The term Workbench refers to the desktop development environment.

Each Workbench contains one or more perspectives. Perspectives contain views and editors and control what appears in certain menus and tool bars.

The following preferences can be changed on the **General** preference page.

Always run in background

Turn this option on to perform long running operations in the background without blocking you from doing other work.

Keep next/previous part dialog open

If this option is turned on then the editor and view cycle dialogs will remain open when their activation key is let go. Normally the dialog closes as soon as the key combination is released. Off

Show Heap Status

Turn this option on to display an indicator showing information about current Java heap usage.

Open mode...

You can select one of the following methods for opening resources:

- Double click - Single clicking on a resource will select it and double clicking on it will open it in an editor.
- Single click (Select on hover) - Hovering the mouse cursor over the resource will select it and clicking on it once will open it in an editor.
- Single click (Open when using arrow keys) - Selecting a resource with the arrow keys will open it in an editor.

Note: Depending on which view has focus, selecting and opening a resource may have different behavior.

Help Content preference page

Help topics from remote servers can be included seamlessly into the local help system. Use the **Help > Content** preference page to configure one or more remote server to include content from.

Include help content from a remote infocenter

If checked, this option enables the use of remote help content. The rest of the fields on the page are only enabled if this option is checked.

Add/Edit/Delete

Add, edit or delete a remote data source

View Properties

View the properties for this remote data source

Test Connection

Tests to see if it is possible to connect to this host/port combination

Disable/Enable

Allows a data source to be disabled so the help system will not try to read topics from that source.

Pressing the "Add" button opens a dialog to add a new infocenter, these are the fields that can be entered in the Add new infocenter dialog.

Name

A name for this infocenter

Host

Specifies the host name of the system that is running the infocenter to serve remote help content. This must be a host name and cannot be a URL (i.e. it cannot start with "http://")

Path

Specifies the context root of the Infocenter application running on the host.

Port

If "Use default port" is selected, port 80 will be used to access remote content on the host. To use any other port the "Use port" option must be selected and the correct port must be specified in the text field.

Keys preference page

The function of the keyboard can be extensively customized in RT-LAB using the **General > Keys** preference page. Within RT-LAB, key strokes and key sequences are assigned to invoke particular commands.

Key Strokes, Key Sequences, and Key Bindings

A 'key stroke' is the pressing of a key on the keyboard, while optionally holding down one or more of these modifier keys: Ctrl, Alt or Shift. For example, holding down Ctrl then pressing A produces the key stroke Ctrl+A. The pressing of the modifier keys themselves do not constitute key strokes.

A 'key sequence' is one or more key strokes. Traditionally, Emacs assigned two or three key stroke key sequences to particular commands. For example, the normal key sequence assigned to Close All in emacs is Ctrl+X Ctrl+C. To enter this key sequence, one presses the key stroke Ctrl+X followed by the key stroke Ctrl+C. While RT-LAB supports key sequences of arbitrary lengths, it is recommended that keyboard shortcuts be four key strokes in length (or less).

A 'key binding' is the assignment of a key sequence to a command.

Schemes

A 'scheme' is a set of bindings. RT-LAB includes three schemes:

- Default
- RT-LAB (extends Default)
- Emacs (extends Default)

The Default scheme contains a general set of bindings, in many cases recognizable as traditional key sequences for well known commands. For instance, Ctrl+A is assigned to Select All, and Ctrl+S is assigned to Save.

It is important to understand why the RT-LAB scheme says that it 'extends Default'. The RT-LAB scheme is not a complete set of bindings like the Default scheme. Rather, it borrows from the Default scheme where possible, only defining explicit RT-LAB-style bindings where they vary from the Default scheme.

Choose the scheme you are most comfortable with by changing the 'Scheme' setting on the keys preference page. If you choose the Default scheme, all RT-LAB bindings are ignored. If you choose the RT-LAB scheme, explicit RT-LAB-style key sequence assignments take precedence over any conflicting assignments in the Default scheme.

Contexts

Key bindings can vary based on the current context of RT-LAB.

Sometimes the active part might be a text editor, for instance, where a different set of key sequence assignments may be more appropriate than if the active part was an html file editor. This context is usually determined by the active part, but it can be influenced by the active window or dialog as well. If the active part does not choose a particular context, the workbench will set the active context to In Windows.

RT-LAB includes a number of different contexts. Some examples are:

- In Dialogs and Windows
- In Windows (extends In Dialogs and Windows)
- In Dialogs (extends In Dialogs and Windows)

- Editing Text (extends In Windows)
- Debugging (extends In Windows)
- In Console

Much like configurations, contexts can extend other contexts.

Note: It is not recommended to promote a key binding to a context which it extends. For example, it is not recommended to move an Editing Text key binding to the In Dialogs and Windows context. This may have unexpected results.

It is possible for some key bindings to work in dialogs. Those key bindings are assigned to the In Dialogs and Windows context. One example of such a key binding is the key binding for "cut". It is possible to change these key bindings. For example, it is possible to have Ctrl+X as cut in dialogs, but Ctrl+W as cut in windows.

Customizing Key bindings

With multi-stroke key sequences, schemes, and contexts, there are a lot of things to keep in mind when customizing key bindings. To make things easier, all key customization is done on the **General > Keys** preference page.

In this example we want to bind CTRL+5 to the About command. By default the keys preference page will show you all possible keybindings. You can see the About command listed in the Help category. You can bind the command by putting focus in the Binding text box and pressing CTRL and 5 like you would if you were executing the command.

When you type CTRL+5 you have created a binding for About. The right-most column will indicate that this is a user binding by displaying a U. If there was a conflict with another key, this column would also display a C. The binding will be in the default context, "In Windows". You can now use the **When combo box** to change the key binding context (for example, to move this binding to "Editing Text").

If you wanted to add a second key binding to About, you can use the **Copy Command** button to create a second command entry for you to bind another key to. If you want to delete a binding, you can either use the **Remove Binding** button or simply give focus to the Binding text box and hit Backspace.

The Dynamic Nature of Key bindings

Key bindings are provided by plug-ins, and in RT-LAB, plug-ins can be added or removed. This can cause key bindings declared by these plug-ins to be added or removed. RT-LAB stores custom key bindings in a way to compensate for this. Consider the example above where CTRL+6 was assigned to About in the Default scheme. Say you install a new plug-in that assigns CTRL+6 to a particular command. RT-LAB will preserve your assignment to About.

Conflict Resolution

There are only a finite number of simple, common key strokes available to assign to a multitude of commands. We have seen that scheme, context and platform all partition key sequence assignments into domains where they don't conflict with one another. Consider the case for Ctrl+B above if contexts did not exist. One plug-in would assign Ctrl+B to Build, the other plug-in would assign Ctrl+B to Make Bold Text. How would RT-LAB properly resolve this conflict?

Though conflicts are drastically reduced by employing the above mechanisms, they can still occur. Two plug-ins, independent of one another, could assign the same key sequence to different commands with the same context, scheme, platform, and locale. Consider if a plug-in assigned Ctrl+F4 in the In Windows context and Default scheme to one of its commands. This directly conflicts with RT-LAB assigning Ctrl+F4 to the close command in the same context and scheme.

This is a conflict. It wouldn't be proper to invoke both commands, nor would it be proper to simply choose one of the two commands to receive the key stroke. We pop up the Key Assist Dialog with the conflicting commands and allow the user to select one. The Key Assist Dialog is the same dialog that displays command choices for multiple key stroke key bindings. For example, if 2 commands were bound to F12 you might see a small dialog with two choices.

If the user sets a keybinding and creates a conflict, the conflicting bindings will be displayed in the conflicts list. This can be used to navigate between conflicting keybindings so that they can be changed.

These types of conflicts can be resolved by explicitly assigning the key sequence to one of the commands, or remove it from the other.

Another type of conflict can be caused by multiple-key stroke key sequences. For example, in the Emacs scheme, there are many multiple-key stroke key sequences beginning with the key stroke Ctrl+X. Ctrl+X K is assigned to Close. Ctrl+X H is assigned to Select All.

As previously mentioned, the Emacs scheme borrows key bindings from the Default scheme. In the default scheme, Ctrl+X is assigned to Cut. Though the Emacs scheme doesn't explicitly redefine Ctrl+X, pressing Ctrl+X is required as part of many of its key bindings. In the Emacs scheme, when one presses Ctrl+X, one is half way to entering one of many possible assigned key sequences. One would not expect the Cut action to be invoked at this time.

For this type of conflict, the rule is that the Ctrl+X key sequence assigned to Cut would be ignored. Otherwise, it would not be possible to complete many of the key bindings in the Emacs configuration.

Label Decorations preference page

Label Decorations allow additional information to be displayed in an item's label and icon.

The **General > Appearance > Label Decorations** preference page provides a description of each decoration and allows the selection of which decorations are visible.

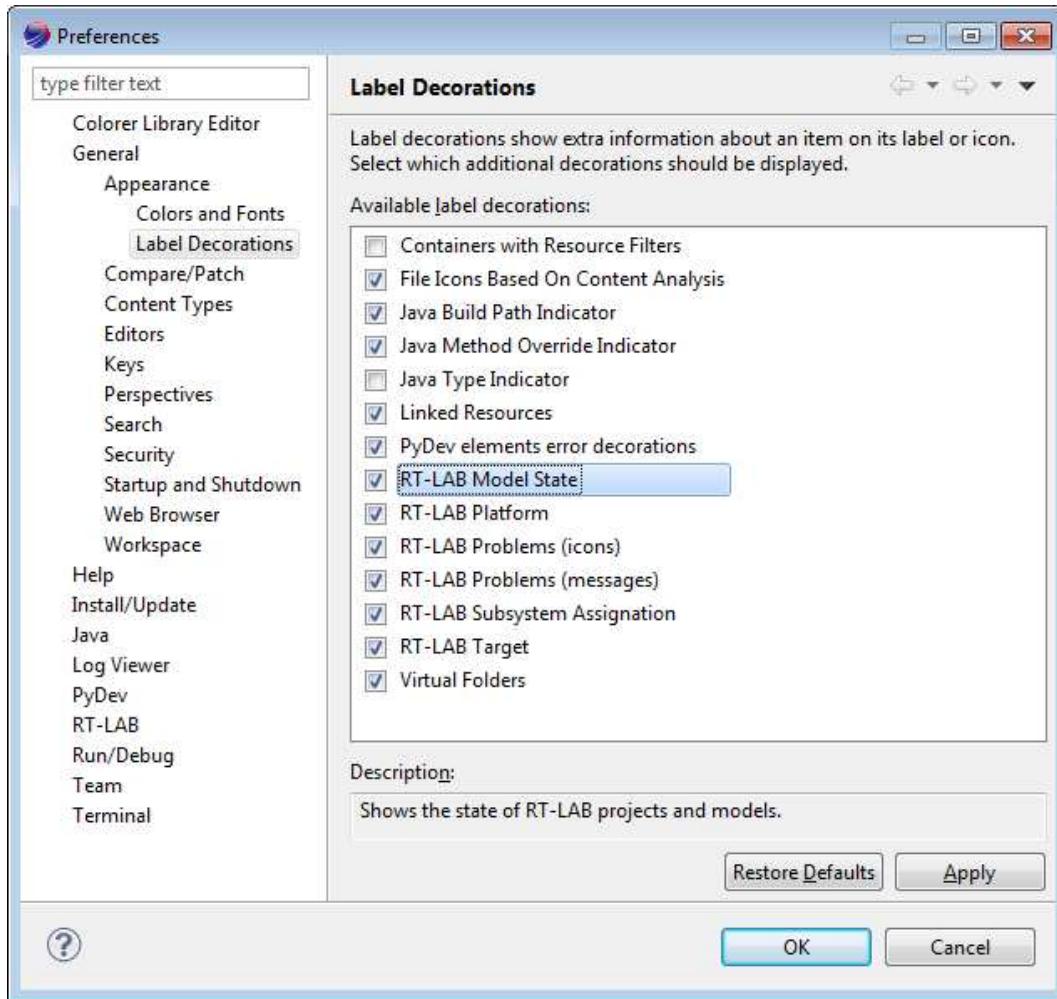


Figure 106:Preference page: Label Decorations

Linked Resources preference page

The **General > Workspace > Linked Resources** preference page is used when working with linked resources. The preference **Enable linked resources** is used to globally enable or disable the linked resource feature for the entire workspace. By default, linked resources are enabled. If you disable linked resources, then you will not be able to create any new linked resources, or import existing projects that contain linked resources.

Not all versions of the workbench support linked resources and recognize them as such. You may not want to use linked resources if you plan to share your workspace data with other users. Disable this preference if they will not be able to work with linked resources.

The remainder of this page is for defining path variables that are used when creating linked resources. Use the **New** button to define new variables, the **Edit** button to change the value of an existing variable, and the **Remove** button to get rid of an existing variable. Note if you change a path variable that is currently in use, you will need to perform a local refresh on those projects to "discover" what is different in the file system. You can refresh a resource by opening the one of the navigation views' context menu for that resource and selecting **Refresh**. It is not recommended that you remove a path variable that is currently in use.

Local History preference page

The following preferences can be changed on the **General > Workspace > Local History** page.

Days to keep files

Indicates for how many days you want to maintain changes in the local history. History state older than this value will be lost.

Maximum entries per File

Indicates how many history states per file you want to maintain in the local history. If you exceed this value, you will lose older history to make room for new history.

Maximum file size (MB)

Indicates the maximum size of individual states in the history store. If a file is over this size, it will not be stored.

Perspectives preference page

On the **General > Perspectives** preference page, you can manage the various perspectives defined in the Workbench.

Open a new perspective

Use this option to set what happens when you open a new perspective. Do you want the perspective opened within the current Workbench window or opened in a new window?

Open a new view

Use this option to specify what happens when a new view is opened. It is either opened to its default position within the current perspective or it is opened as a fast view and docked to the side of the current perspective.

New project options

Use this option to specify the perspective behavior when a new project is created. You can set it to switch the current perspective to be the one associated with the project type and open the perspective in the same Workbench window as the current one, switch the perspective and open it in a new Workbench window, or not to switch perspectives at all.

Available Perspectives Options:

Make Default

Sets the selected perspective as the default perspective.

Reset

Resets the definition of the selected perspective to the default configuration. This option is only applicable to built-in perspectives that have been overwritten using **Window > Save Perspective As...**

Delete

Deletes the selected perspective. This option is only applicable to user-defined perspectives (built-in perspectives can not be deleted).

Quick Diff preference page

The following preferences can be changed on the **General > Editors > Text Editors > Quick Diff** preference page.

Enable quick diff

This option will enable or disable the quick diff option.

Show differences in overview ruler

This option will show differences in the overview ruler.

Colors - Changes

This option controls the color of changes.

Colors - Additions

This option controls the color of additions.

Colors - Deletions

This option controls the color of deletions.

Use this reference source

This option sets which reference to use as the base for generating quick diff comparisons. Options are:

- **Version on Disk:** Current file is compared against the last saved version on disk.

Search preference page

The **General > Search** preference page allows you to set preferences for searches.

Reuse editors to show matches

This option allows you to keep using the same editor for search results to reduce the number of open editors.

Bring Search view to front after search

This option will display the search view at the front after performing a search.

Ignore potential matches

Select this option if you only want to see exact matches.

Emphasize potential matches

This option allows you to highlight potential matches in the Search view. If the Search engine isn't 100% sure about the match then it is considered a potential match.

Foreground color for potential matches

This option allows you to select the foreground color for potential matches.

Default perspective for the Search view

This option allows you to define which perspective should be brought to the front when there are new search results.

Spelling preference page

This service is not installed with RT-LAB.

Startup and Shutdown preference page

The **General > Startup and Shutdown** preference page allows the selection of plug-ins to be automatically activated during workbench startup.

Normally plug-ins are not activated until they are needed. However some plug-ins may specify that they wish to be activated during startup. This preference page allows the selection of which of these plug-ins will actually be activated during startup.

Prompt for workspace on startup

If this option is turned on then the workbench will prompt you each time it is started for what workspace to use.

Refresh workspace on startup

If this option is turned on then the workbench will synchronize its contents with the file system on startup.

Confirm exit when closing last window

If this option is turned on then the workbench will ask if you wish to exit when closing the last window if.

Plug-ins activated on startup

This option allows you to select which available plug-ins should be activated on startup.

Text Editors preference page

The following preferences can be changed on the **General > Editors > Text Editors** page.

Appearance options

Undo history size

This option allows you to set the size of the undo history for text editors.

Displayed tab width

This option allows you to set the displayed tab width for text editors.

Insert spaces for tabs

This option allows you to insert space characters in place of tab characters.

Highlight current line

This option controls whether or the current line is highlighted or not.

Show print margin

This option controls whether the print margin is visible or not.

Print margin column

This option allows you to set the print margin column position.

Show line numbers

This option controls whether or not line numbers are shown on the left side of the text editor.

Show range indicator

This option controls whether or not range indicators are shown in the text editor.

Show whitespace characters

This option controls whether to display whitespace characters in text editors.

Enable drag and drop of text

This option controls whether text drag and drop is enabled.

Warn before editing a derived file

This option controls whether to warn if a derived file is going to be edited.

Smart caret positioning at line start and end

This option controls whether the editor automatically positions the caret and the start or end of a line.

Show affordance in hover on how to make it sticky

This option controls whether to show an affordance in the hover on how to make it sticky.

Appearance color options

This option controls various appearance colors.

Web Browser preference page

The following preferences can be changed on the **General > Web Browser** preference page.

Use internal Web browser

This option enables you to use an internal Web browser.

Use external Web browser

This option enables you to use an external Web browser. Select the required browser from the list of available external web browsers.

Workspace preference page

On the **General > Workspace** preference page, you can manage various IDE-specific workspace preferences settings in the Workbench.

Build automatically

If this option is turned on, then the Workbench will perform an automatic build whenever a modified resource is saved.

Save automatically before build

If this option is selected, when a manual build is performed the Workbench will automatically save all resources that have been modified since the last build was performed.

Workspace save interval (in minutes)

This number indicates how often the state of the workspace is automatically saved to disk.

Refresh automatically

If this option is turned on then the workspace resources will be synchronized with their corresponding resources in the file system automatically.

Note: This can potentially be a lengthy operation depending on the number of resources you have in your workspace.

Open referenced projects when a project is opened

If this option is enabled opening a project will also open and closed projects it references. Select prompt if you wish to be asked first.

Text file encoding

Use this option to specify the encoding to use when saving text files in editors.

Text File line delimiter

Use this option to specify the line delimiter to use for new text files. Note: This will generally not effect the file line delimiter for existing files.

Cheat Sheets

RT-LAB provides cheat sheets to guide you through some of its application processes. Each cheat sheet is designed to help you complete some task, and it lists the sequence of steps required to help you achieve that goal. As you progress from one step to the next, the cheat sheet will automatically launch the required tools for you. If there is a manual step in the process, the step will tell you to perform the task and click a button in the cheat sheet to move on to the next step. Also, relevant help information to guide you through a task is retrieved in a single click so that lengthy documentation searches will no longer be required.

Launching a cheat sheet

To launch a cheat sheet from the Workbench

- Select **Help > Cheat Sheets** from the menu bar.
- The available cheat sheets are listed, select one and click **OK**. Alternatively if you know the location of a cheat sheet content file on your file system or on the web you can enter its path.

The cheat sheet opens as a view. At any time, only one cheat sheet is open and active. When you launch a cheat sheet, any opened cheat sheet is closed before the new one is opened. The completion status of closed cheat sheet is saved.

The cheat sheet has a toolbar at the top right edge. These icons appear in the toolbar:

ICON	DESCRIPTION
Collapses all	Collapses all the expanded steps except the current step or expands steps to the last expanded state. Click to toggle between these two states.
Select	Allows you to select and open another cheat sheet. The completion status of the active cheat sheet is saved. Then, the active cheat sheet is closed and the selected cheat sheet is opened.
Hides	Hides the cheat sheet.
Save	Saves the completion status of the active cheat sheet and closes it.

Saves the completion status of the active cheat sheet and closes it.

Note that some cheat sheets can also be launched from the welcome page by first selecting Tutorials and then selecting one of the tutorials.

Starting the cheat sheet

Each cheat sheet has a list of steps and it always begins with an Introduction step. When you launch a fresh cheat sheet, the Introduction step is expanded so that you can read a brief description of the cheat sheet. To start working with the cheat sheet, click **Click to Begin** in that step. The next step is expanded and highlighted. You should also see one or more actions buttons, such as **Click to Perform** in the highlighted step. You can now begin working through the tasks using the cheat sheet. At any time, the only highlighted step in the cheat sheet is the current step.

Restarting the cheat sheet

Any time after starting a cheat sheet, you can restart from the first step by clicking **Click to Restart** in the Introduction step. If you have already created some artifacts, you will have to manually clean up the workspace before restarting the cheat sheet.

Progressing through the steps

In the current step, when you click **Click to Perform**, a tool (which can be a wizard), will be launched and you will be required to work with that tool. When you finish working with that tool, the next step is automatically highlighted and it becomes the current step. When the current step is a manual task, you will need to perform the work and click **Click to Complete** to move to the next step. A check mark appears in the left margin of each completed step.

Getting help information for tasks

To get step-by-step instructions for that step, click the help link in the step before you click **Click to Perform**, and the step-by-step instructions on how to work with that tool will be displayed in the Help window.

Additional help for entry fields in the tool or wizard may be available by focusing on the field (use the Tab key to position to that entry) and pressing F1.

Skippping a step

If a current step has a **Click to Skip** option, then it is an optional step. You must click **Click to Skip** to skip the current step, when you do, the step will have the skip mark in the left margin. If the task does not present **Click to Skip**, you must perform that step and you cannot skip it.

Redoing a step

You can redo any step that you may have completed or skipped in the current cheat sheet. To redo the step, expand the step by clicking its expand icon and then clicking **Click to Redo**. After redoing a step, the cheat sheet will continue from the redo step.

Closing the cheat sheet

When you finish the last step in a cheat sheet, it automatically restarts. You can also close the active cheat sheet by clicking the close icon in the cheat sheet's toolbar. The active cheat sheet saves its completion status when it is closed so that you can continue where you left off at a later time.

MetaController

The **MetaController** is a server required by RT-LAB to interact with the RT-LAB simulator. This application is started when the operating system starts and runs in the background. The **MetaController** also helps you to launch all applications related to RT-LAB that interact with the simulator.

Note that the **MetaController** could be also automatically launched when the RT-LAB user interface is started.

Icon system tray

On Windows command station, the **MetaController** is shown in the system tray located in the Windows Taskbar, usually at the bottom right corner next to the clock. See the following figure.

The system tray icon provides various useful features such as starting the RT-LAB user interface. When you click on it with your right mouse button, a popup menu will appear with the available features.

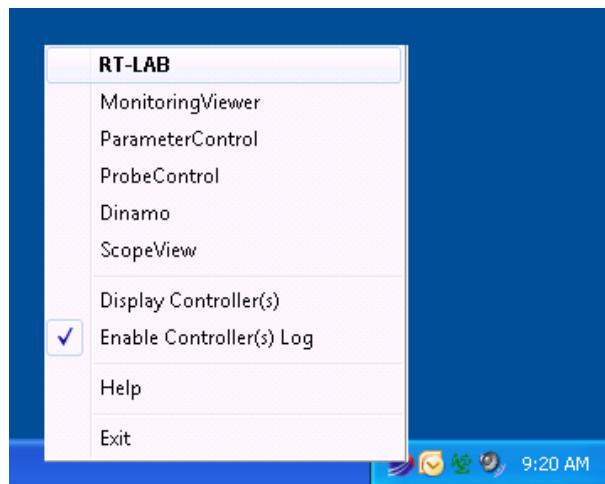


Figure 107:MetaController icon system tray (Windows only)

Menus

Here are descriptions of the **MetaController** menus:

- **RT-LAB:** Start the RT-LAB **Workbench**. Double-clicking on the icon in the system tray will also start the RT-LAB user interface.
- **MonitoringViewer:** Starts the **Monitoring View**.
- **ProbeControl:** Starts the **Probe Control Panel**
- **Dinamo:** Starts the DINAMO panel.
- **ScopeView:** Starts the ScopeView application to acquire and display signals from real-time simulation. See [Using ScopeView](#) for more information.
- **Display Controller(s):** Displays RT-LAB Controller application in the taskbar. Unchecked by default.
- **Enable Controller(s) log:** Creates a file called Controller.log in the RTLAB_ROOT\common\bin folder for debugging use. Unchecked by default.
- **Help:** Opens the RT-LAB Help Center.
- **Exit:** Closes MetaController.

Help

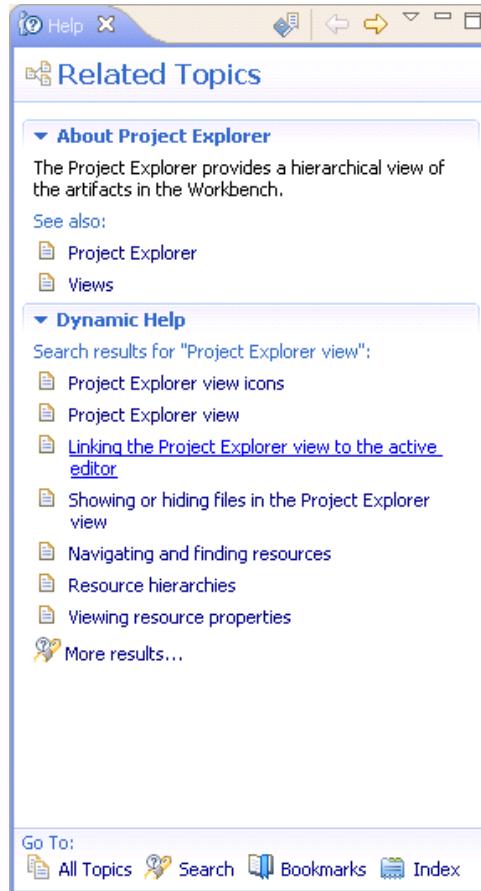
The Help system lets you browse, search, bookmark, and print help documentation. The documentation is organized into sets of information that are analogous to books. The help system also supplies a text search capability for finding the information you need by search phrase or keyword, and context-sensitive help for finding information to describe the particular function you are working with.

You can interact with help system in the workbench using the Help view or in the separate Help window. The view and window provide the same information but in different ways.

The Help view

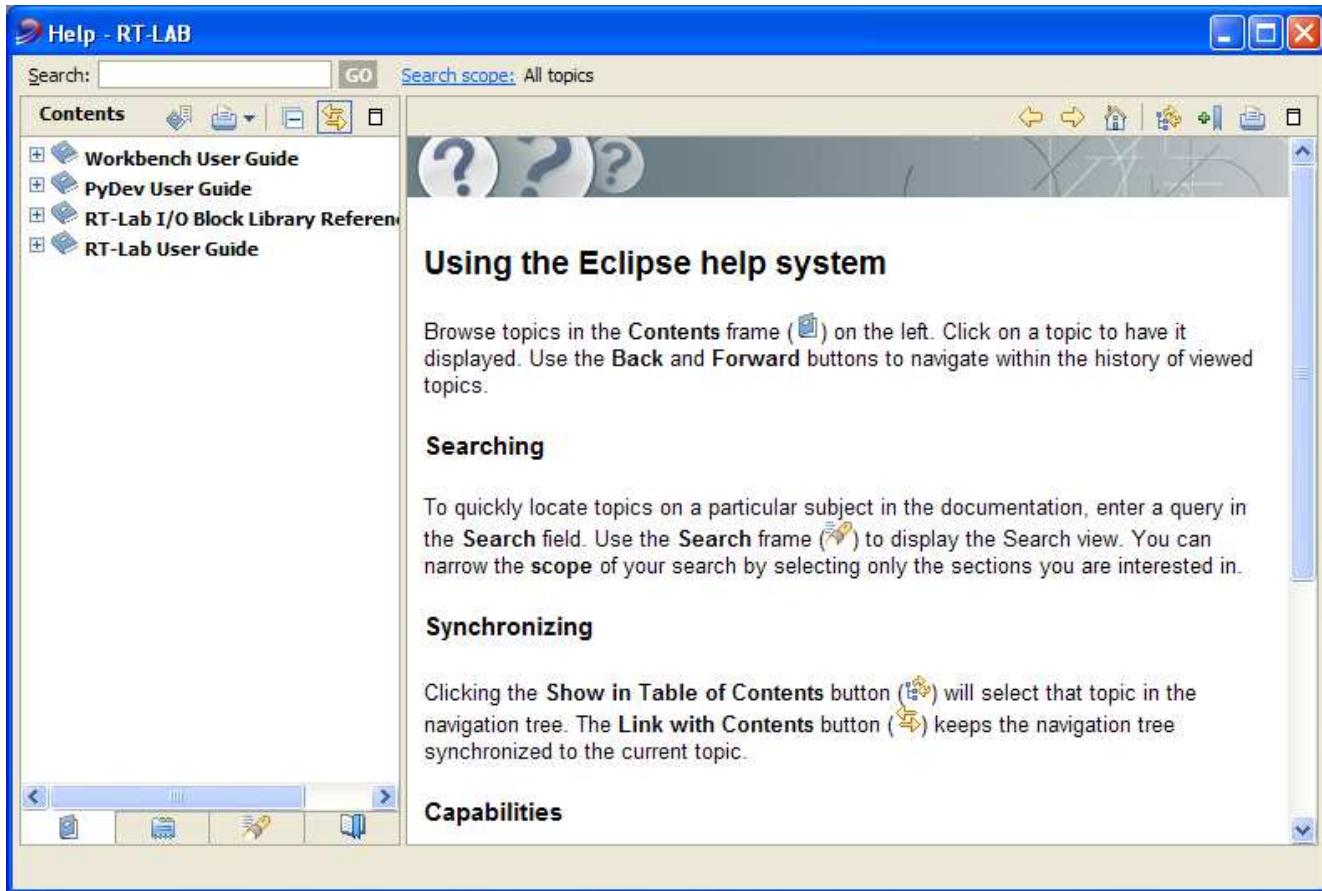
The Help view provides help inside the workbench. You can open the view from the main menu by selecting **Help > Dynamic Help** or **Help > Search**. The view will open showing the Related Topics or Search page, respectively. You can use links at the bottom of the help view to turn to other pages.

You can read more about the [Help view](#) and its various pages.



The Help window

The Help window provides the same content as the Help view, but in a separate window instead of in a view. You can open the window from the main menu by selecting **Help > Help Contents**. The first view shown in the window is called Contents. This view displays the table of contents for the product documentation. Click on one of the links to expand the navigation tree for a set of documentation.



Context-sensitive help

If you are working through a task and encounter a part of the interface that you do not understand, you can summon context-sensitive help. By default, this will display the Help view and give you some specific information about the view/editor/dialog you are using, and possibly some links to topics for further help.

Context-sensitive help can be accessed by bringing focus to the interface part in question by clicking on it or using the Tab key, and then pressing F1. Alternatively, in dialogs you can achieve the same result by pressing the help button in the dialog's button bar.

Related concepts

[Help view](#)

Tasks

Contents:

- [**Building models**](#)
- [**Executing models**](#)
- [**Acquiring and Viewing Data**](#)
- [**Monitoring Models**](#)
- [**Using RT-LAB Blocks**](#)
- [**Using ScopeView**](#)
- [**RT-LAB Connectivity**](#)
- [**Embedding Simulation**](#)
- [**Taking a Snapshot**](#)
- [**Working with perspectives**](#)
- [**Working with views and editors**](#)
- [**Customizing the Workbench**](#)

Building models

This chapter describes the basics of RT-LAB, provides an overview of the simulation process, from designing and validating the model, to using block diagrams and I/O devices, to running the simulation and using the Console as a graphic interface.

Building a simple model for RT-LAB

Naming convention

In RT-LAB, all top-level subsystems must be named with a prefix identifying their function. The prefixes are:

- SC_: console subsystem. The Console subsystem is the subsystem operating on the command station that enables you to interact with the system. It contains all the Simulink blocks related to acquiring and viewing data (scope, manual switch, To Workspace-type blocks, etc.). The blocks you need, whether it is during or after the execution of the real-time model, must be included in the Console subsystem. The console runs asynchronously from the other subsystems. Note that there can only be one Console per model.

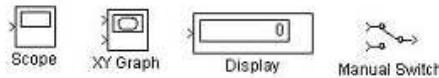


Figure 1:Simulink Blocks for Console Subsystem

- SM_: master subsystem. There is always one and only one master subsystem inside a model. It contains the computational elements of the model.
- SS_: slave subsystem(s). There can be several slave subsystems inside the model. It contains the computational elements of the model when distributing the processing across multiple nodes.

These top-level subsystems map directly to logical nodes. The SC subsystem is the command station while SM is the target. For distributed computation, the SS subsystems allow you to dispatch additional computation on other targets. Each CPU of a target (physical node) is considered a logical node.

- 1 target: SM subsystem only
- 2 targets: SM and SS subsystems
- 1 target (2 CPU): SM and SS subsystems
- 2 target (2 CPU each): SM and 3 x SS subsystems

Native Simulink model

RT-LAB supports any Simulink model without modifying the model. However, this model has to be compatible with Real-Time Workshop (RTW) before opening it with RT-LAB:

- uses fixed step solver;
- uses blocksets compatible with RTW.

During the compilation process, the model will be automatically separated (no modification are done to the original model) into one computation subsystem and one console with one acquisition group. The signals to display could be added later dynamically when the model is running. Interaction to the model is done by setting the model's parameter on-the-fly.

Single subsystem model

You may want to modify your model to make it compatible with RT-LAB. It could be useful when you want to set some signals to be always sent from the SM to the SC subsystem and also to define some control signals. In this case, the rules for creating the model is the same than a distributed model except that there will be no SS subsystems. Refer to the next paragraph to learn how to create an RT-LAB model.

Building a distributed Model for RT-LAB

Any Simulink model can be implemented in RT-LAB, but some modifications must be made to distribute the model and transfer it into the simulation environment. The success of distributed computing with a complex model will depend on the separation of that model into small subsystems synchronized to run in parallel. This should be kept in mind early in and throughout the design process.

To build an distributed model, users must modify the block diagram (**.mdl** or **.sbd** file) by regrouping the model into calculation subsystems, inserting OpComm communication blocks, taking into account step size calculations and exchanging state variables or using Delay blocks; each of these topics is covered within this chapter. After these steps have been completed, RT-LAB begins the compilation process with the regrouped file, separating the model and generating and compiling the code. You then sets execution settings, which are covered in the following chapters, after which point the model's simulation is ready to be executed.

This chapter describes the steps necessary to build an RT-LAB model by modifying your Simulink block diagram to ensure it is properly separated for distributed execution and to maximize its performance.

RTdemo2 example

This model consists of:

- Plant: The physical system being controlled.
- Controller: A device that receives data from the sensors and, based on the results of a control algorithm calculation, sends control signals to actuators so as to induce desired behavior in the plant.
- Human interface: Provides for human interaction with the system

This logical separation will be used to create the top-level subsystems

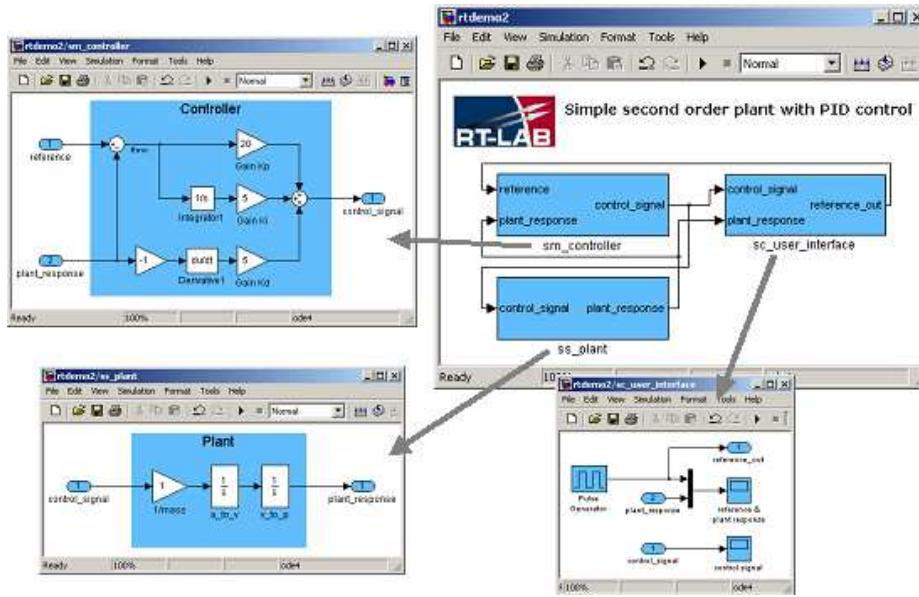


Figure 2:rtdemo2 model

OpComm communication blocks

Once the model is grouped into Console and computation subsystems, special blocks called **OpComm** blocks must be inserted into the subsystems. These are simple feed-through blocks that intercept all incoming signals before sending them to computation blocks within a given subsystem. **OpComm** blocks serve several purposes:

- When a simulation model runs in the RT-LAB environment, all connections between the main subsystems (SC_{_}, SM_{_}, or SS_{_}) are replaced by hardware communication links. For communication between real-time target nodes (SM_{_} and SS_{_}) RT-LAB uses a designated real-time link. For communication between the Console (SC_{_}) and the real-time nodes (SM_{_} or SS_{_}) RT-LAB uses TCP/IP.

Because of these communication links between nodes, the simulation may not run the same way in Simulink as in RT-LAB. In RT-LAB, a computation subsystem waits for reception of all signals before it is able to start calculating. In Simulink, on the other hand, computations performed on a signal start as soon as the signal is available. As pass-through icons that wait for all inputs before making their outputs available, **OpComm** blocks emulate the behavior of the system as it is run in RT-LAB. Keep this effect in mind when designing models.

If you receive a message indicating that the inserted OpComm blocks create an algebraic loop, this means your model cannot be run in real-time, as a deadlock has been created. To correct this problem, include a state variable. See the full discussion in section Maximizing Parallel Communication.

- **OpComm** blocks provide information to RT-LAB concerning the type and size of the signals being sent from one subsystem to another.
- **OpComm** blocks inserted into the Console subsystem enable you to select the data acquisition group you want to use to acquire data from the model and to specify acquisition parameters. (Acquisition parameters are described in the following sections).

RT-LAB uses OpComm blocks to enable and to save communication setup information. This includes both communication between the command station and computation nodes and communication between computation nodes in a distributed simulation scenario.

All inputs to top-level subsystems must first go through an OpComm block before they can be used.

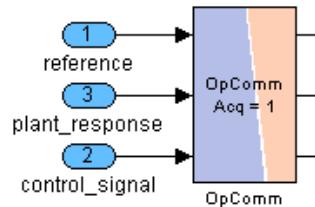


Figure 3:OpComm block

OpComm placement rules

In the computation subsystems (SM or SS):

- One OpComm receives real-time-synchronized signals from other computation subsystems
- One OpComm receives signals asynchronously from the console subsystem

In the console subsystem (SC subsystem):

- One or more OpComm blocks may be inserted to receive signals from the computation nodes. Multiple OpComm blocks define unique “acquisition groups” with their own data acquisition parameters

OpComm blocks in the SM subsystem

One OpComm block is inserted to receive signals from the SC subsystem (asynchronous network).

One OpComm block is inserted to receive signals from other real-time subsystems (in this case, plant_response, from SS)

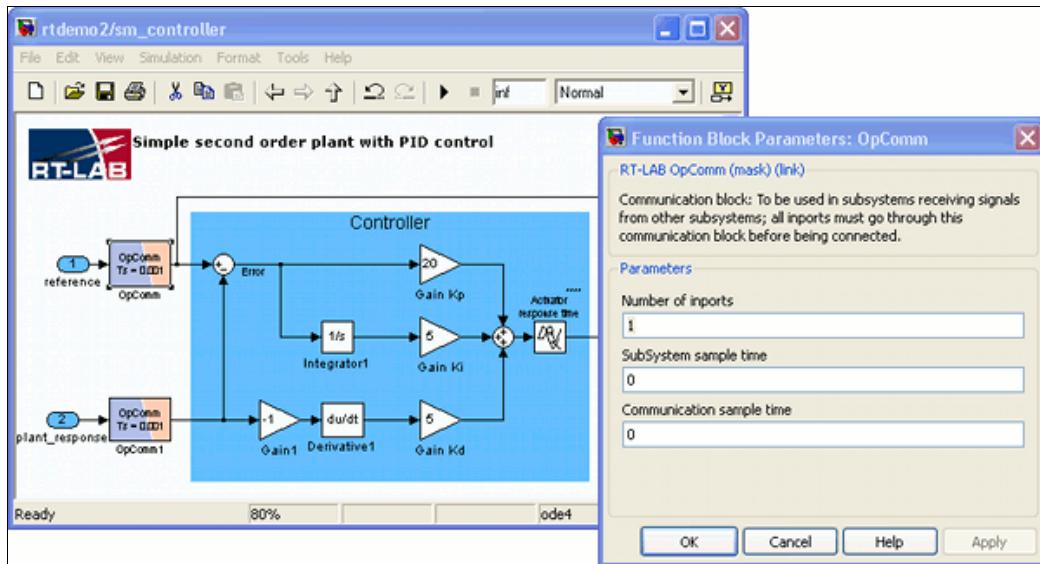


Figure 4:SM subsystem and associated OpComm mask

OpComm blocks in the SS subsystem

Since data are only from the SM subsystem, only one OpComm block is inserted into the slave subsystem.

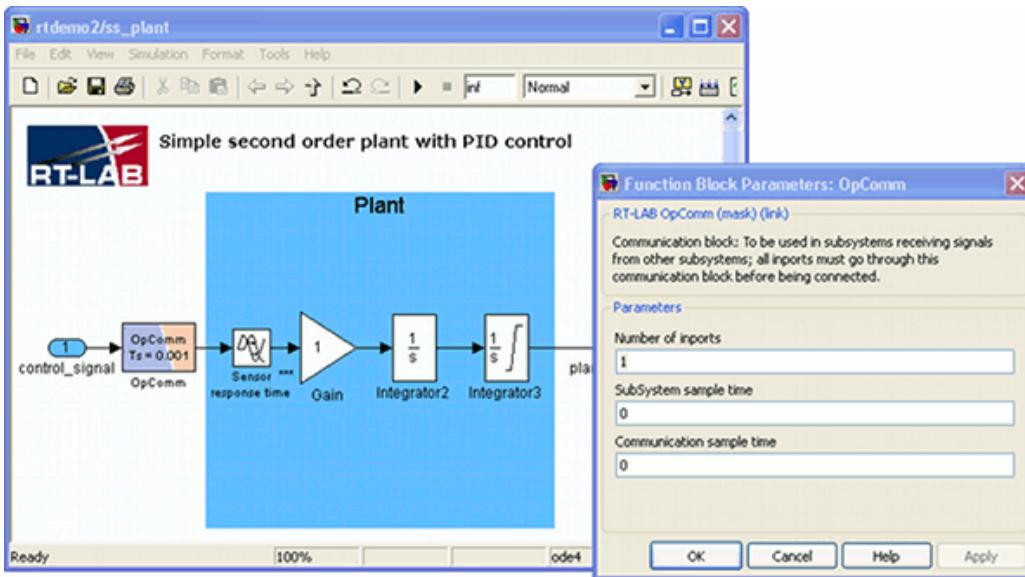


Figure 5:SS subsystem and associated OpComm mask

OpComm blocks in the SC subsystem

For the console, 1 or many OpComm blocks can be added depending on how many acquisition groups you want to set. Acquisition groups allow you to define different priorities and acquisition parameters to different groups of signals.

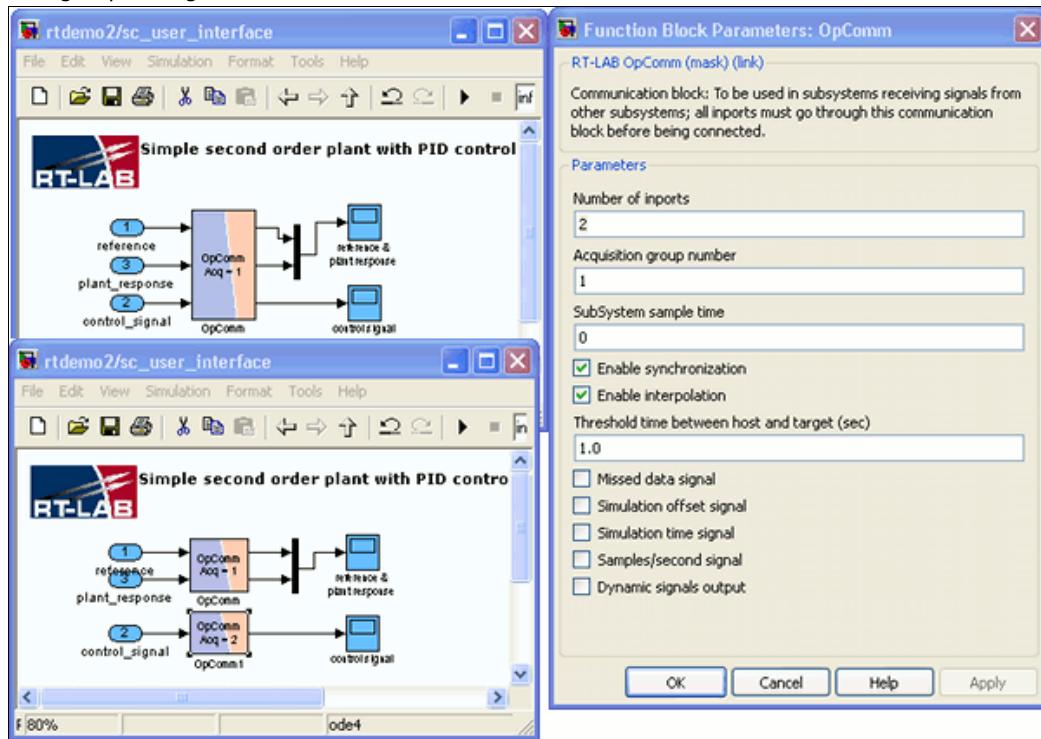


Figure 6:SC subsystem and associated OpComm block

Since the SC (console) subsystem run asynchronously from the main computation (SM and SS), following limitations applies:

- The console subsystem cannot contain computation on which the real-time model relies
- Only signals from the same acquisition group (OpComm) can be compared with each other. Signals from different OpComm blocks are not synchronized together

Maximizing Parallel Communication

To maximize parallel communication, output data must be exchanged between your network's nodes in the most efficient way possible. The order in which data is sent can be prioritized so that nodes run their main calculations without having to wait for outputs from other nodes in the network. This is accomplished by identifying calculations that can be run independent of inputs as state variables and passing input-dependant variables through **Delay** blocks.

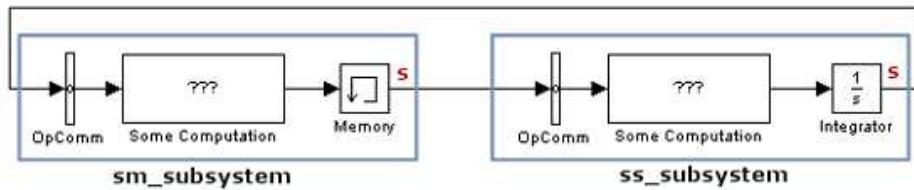
State variable

A state can be defined as an output (signal) which is computed only from preceding inputs or outputs. Example of blocks which introduce a state are the "integrator" and "memory" blocks.

A "gain" block does not produce a state because its output at step z depends on its input at the same step.

Distributed examples

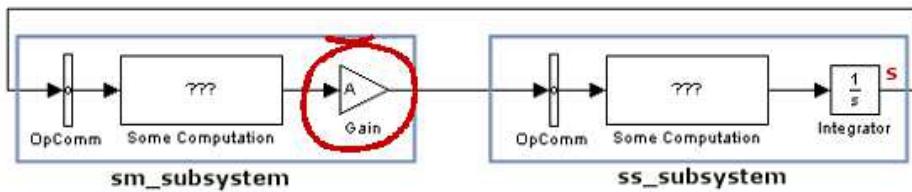
Fully parallel execution



At each step, RT-LAB does the following:

1. **ss_subsystem** sends to **sm_subsystem**.
2. **sm_subsystem** sends to **ss_subsystem**.
3. computation of both subsystems.

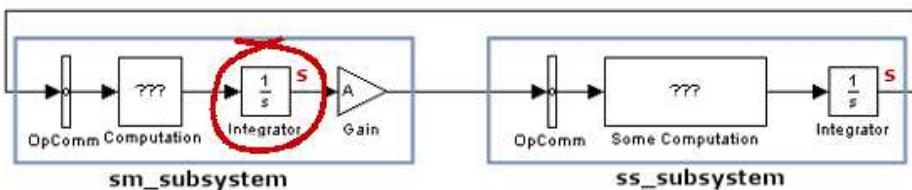
Serial execution (worst case)



At each step, RT-LAB does the following:

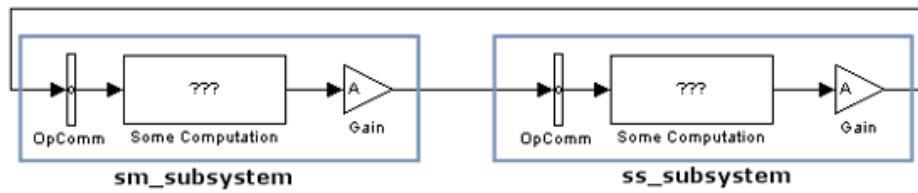
1. **ss_subsystem** sends to **sm_subsystem**.
2. computation of **sm_subsystem**.
3. **sm_subsystem** sends to **ss_subsystem**.
4. computation of **ss_subsystem**.

Part parallel execution (intermediate case)



At each step, RT-LAB does the following:

1. **ss_subsystem** sends to **sm_subsystem**.
2. computation of the gain in **sm_subsystem**.
3. **sm_subsystem** sends to **ss_subsystem**.
4. computation of **ss_subsystem** and the rest of **sm_subsystem**.

Deadlock case

RT-LAB is deadlocked!

1. **sm_subsystem** waits for **ss_subsystem**.
2. **ss_subsystem** waits for **sm_subsystem**.

Executing models

This chapter describes the simulation types available with RT-LAB, explains how to build and execute a model.

Target platform

RT-LAB supports three different target platforms:

- Windows XP/Vista/7: For pure simulation or pseudo real-time execution.
- QNX 6.x: For pure simulation or real-time execution.
- Red Hat Linux: For pure simulation or real-time execution.

Simulation mode

Standard mode

- **Simulation:** Free-run, as-fast-as-possible simulation on target. In this mode, no synchronization is done. Target nodes in a distributed simulation are synchronized with the communication link. The data exchanged depends on the model's data flow.
- **Simulation in low priority (Win32 target only):** Same as simulation but the computation subsystem runs in lower priority. It is useful when the simulation runs on the command station; it will let cpu resource to other applications.
- **Software synchronized:** the model runs in real-time. In this mode, the model is synchronized on the internal RTOS timer. Only one computation node is synchronized when executing a distributed simulation. Other subsystems are synchronized with the communication link. The data exchanged depends on the model's data flow.
- **Hardware synchronized:** the model runs in real-time. In this mode, the model is synchronized on an external hardware timer. A specific block is required to be inserted in the model to specify and configure where the external clock is located. Only one computation node is synchronized when executing a distributed simulation. Other subsystems are synchronized with the communication link. The data exchanged depends on the model's data flow.

eXtreme High Performance (XHP) mode

The XHP mode is a means for the real-time operating system to disable interrupts. This is done on a per-subsystem (thus per-CPU) basis. Not allowing interrupts prevents process switching which removes latencies time for additional computation in the same time slice.

The XHP mode is aimed at real-time applications of a given base sample time that cause overruns (overflow their allowed time-step for computation) while running in RT-LAB in the standard mode.

In XHP mode, the model waits in an empty loop for its next scheduled computation step. The next model step is then computed.

In this mode, we use the CPU counter as our time reference. Because this counter operates at the CPU's frequency, it offers a very high resolution, even for step-sizes in the microsecond range. Because the counter resides on the CPU and reading its value is done within one CPU cycle, this operation introduces almost no latency.

Issues Introduced by XHP Mode on QNX target

On QNX target, the XHP mode is enabled on the whole computer. All the subsystem running on this same target will run in XHP. It means that the number of subsystems that could run on this target depends on the number of CPU since each subsystem will be assigned on a specific CPU. Because the

computation subsystem is the only process running on the processor when using XHP mode, some restrictions are introduced by its use. As a consequence of the very low step sizes attainable by subsystems in XHP mode, other background tasks on the computation node will not be executed.

Issues Introduced by XHP Mode on Red Hat Linux target

On Red Hat Linux target, the XHP mode is enabled per cpu basis. It means that only the CPU assigned to an XHP subsystem will not be available to other process and OS. If the number of subsystems running in XHP mode on a SMP target is less than the number of CPUs, the OS will not be disabled and all the functionalities provided by the OS will be available. In this case, the following paragraph explaining limitations for XHP mode will not apply. If the number of subsystems running in XHP mode is equal to the number of CPUs, the target will be set in "full" XHP model and all limitations will apply.

Limitations and Solutions for XHP Mode

Acquisition and model control from the command station

Because the network card driver is not available during XHP runs, the communication between this computation node and the command station is not available. Only the inter-computation-node communication using Firewire or shared memory (on SMP machines) is still available. This means that signals you wish to be acquired during execution in XHP mode must be routed to another, non-XHP, computation node, which will be able to send these signals to the command station. If you do not have a second computation node to use as a relay, you can also set your model to pause automatically (using the RT-LAB "Pause", a "Clock" and some logic blocks) at a given interval. When the model pauses, it switches back to normal mode and the acquired signals can be sent to the command station. Be sure to set your acquisition window to the size of your execution interval. Be aware that this is only possible for models that can be interrupted regularly and is rarely a viable solution because of this.

Distributed computation using UDP/IP communication.

Again, because the network card driver is not available during XHP runs, communication protocols which use this card are not possible. You need to use (OHCI) Firewire cards for communication or multi-processor machines which communicate through shared-memory.

Multi-rate models cannot be executed in "true" multi-rate

Because the ability to maintain multiple threads is disabled in the XHP mode for increased performance, it is not possible to run a multi-rate model in multi-tasking form ("true" multi-rate). Multi-rate systems can be still executed in single-tasking mode, but this has many disadvantages. See more about "true" vs. "pseudo" multirate for details or the Real-Time Workshop User's Guide.

Parameter tuning

Just as for acquisition, parameter modification is not possible for parameters of subsystems running in XHP mode. If you want to be able to modify some parameters, they will have to be relocated on a non-XHP node and wired to the XHP subsystem at the diagram level or while the model is paused.

Snapshot and dynamic signal acquisition

Also because of the single thread restriction, no snapshot can be taken and no signal can be dynamically selected when a subsystem is running in XHP mode. You can, however, pause the model to perform these functions and then continue execution.

Some I/O interfaces are not available

Because system interrupts and multi-tasking are disabled for computation nodes running in XHP mode, any I/O boards which require these features are not available during XHP runs. These blocks are mainly

limited to serial interfaces (which require an asynchronous process) and discrete event detectors/generators (which require interrupts).

Using XHP Mode

The XHP mode can only be applied at the subsystem level. To enable the XHP mode, you must check the XHP box in [Assignment Page](#) from [Model Editor](#). On QNX target, if a non-XHP subsystem is assigned physically to the same computer as an XHP subsystem, the computer is set to XHP mode.

Subsystems running in XHP mode won't be able to communicate using the Ethernet network. They will only be able to communicate with other subsystems using the OHCI (IEEE-1394) or Infiniband connections.

It is recommended that you use the XHP mode only on slave subsystems so that the master can relay the communications between the slave nodes running in XHP and the command station (host computer) through Ethernet (or other TCP/IP transport).

An interesting point in having subsystem running in XHP is that it enables you to use a Multi-Processor computer on QNX. Once you enabled the Multi-Processor feature of QNX, you can assign a XHP slave on each processor of that computer.

Communication type

Two computation subsystems on the same physical node (regardless of the number of logical nodes) will exchange real-time signals using a shared memory.

If computation subsystems (SM or SS) are on different physical nodes, you must select which real-time network type to use, either UDP/IP (slow), Firewire (a.k.a. OHCI, IEEE-1394) or INFINIBAND (Red Hat Linux target only).

The real-time link is configured using [Model Editor](#) in the [Execution Page](#).

UDP/IP

This link requires ethernet board on targets. It is a slow link and it should not be used for hard real-time simulation.

OHCI

This link requires OHCI board on target. RT-LAB support OHCI 400 (IEEE 1394a) and OHCI 800 (IEEE 1394b). The type of card is detected automatically. One OHCI board is required per target. Additional card could be added to extend the bandwidth (computation subsystem exchanging data using 2 OHCI boards per target) or to allow multiple subsystems to communicate to other subsystems (when there are multiple subsystems on the same physical nodes, only one subsystem could access an associated OHCI board).

Dophin

This link requires one Dolphin D352 board to be installed on each target. The D352 offers two bi-directional links of 20Gbits/s. The cards' SCI links are hot-swappable connections. It should be used when high data throughput with very low latency, increased failover performance and fault tolerance are needed. For now, support for this communication type is only available on Red Hat.

INFINIBAND

This link requires INFINIBAND board on target. This link should be used when a large bandwidth is required between nodes. It is only available on Red Hat Linux target. Two computation nodes could

communicate by connecting the peer-to-peer the two INFINIBAND board. When there are more than two computation nodes in a distributed simulation, an INFINIBAND switch is required to connect all the targets together.

Executing steps

Compilation

- Select your target platform.
- Start the compilation. This is an automated process, so no user interaction is needed. Here is the list of the steps that are done when compiling a model:
 1. Separation: it is where the original model is separated into as many separate models as there are top-level subsystems.
 1. Code generation: for each computation subsystem (SM and SS), Real-Time Workshop or Autocode is called to generate the C code for RT-LAB using a set of specific templates specific to the target OS.
 1. Transfer to RTOS: the newly generated C files are transferred to one of the computation nodes (called the development or compilation node) via the FTP protocol.
 1. Compilation: for each computation subsystem (SM and SS), the compiler will compile and link the C code into an executable file for RT-LAB.
 1. Retrieval of Executables: retrieve the executable(s) via FTP. This allows RT-LAB to load a subsystem on a different node than the compilation node.

Execution

- Assign the subsystems to physical nodes. You can assign more than one subsystem to one physical node. If there are more than one processor (logical node), then the load will automatically be distributed among the processors. If there is only one processor, then RT-LAB will rely on the RTOS to perform the task switching between the subsystem processes. This is about equivalent to not distributing the model.
- Select “Handle console automatically” from **Simulation Tools Page** from **Model Editor** if you want to use the user interface based on the original model’s SC subsystem.
- Select a simulation mode.
- Configure the communication type (optional).
- Load the subsystem on the computation node(s). The executables (one per computation subsystem) are transferred to their assigned nodes using the FTP protocol. The subsystems are then loaded and ready to run. Note that only one model per target is allowed by default to ensure real-time integrity (QNX and Red Hat only). This option could be disabled by creating a file `/usr/opalrt/multimodels` on the target where you want to enable multiple running models at the same time.
- Execute the model. The simulation is started. The RT-LAB generated console starts displaying the data retrieved. Note that it is NOT the original model, just the SC part of it. You could also create your own UI using the our API instead.
- At this point, you could modify parameters, log data to files...
- Reset the model when your simulation is completed. The model could also reset itself by inserting a stop block inside it. RT-LAB automatically transfers all generated files to the command station to be reviewed later.

Additional files

Some simulation requires additional files in order to compile and execute the model. Refer to [Files Page](#) to see how to set up new files.

Usually header files and source files to be compiled are transferred during the compilation. Data files to be used at run-time are usually transferred while loading the model.

User script files

Since many different configurations may be required in order to run a simulation, RT-LAB supports script calls during the compilation and execution of a model. Scripts, created using Python, can be global to a node or local to a subsystem, called on a command station or target, and are available during the compilation step and the load/reset step. In addition, script outputs can be viewed thru the RT-LAB display panel.

The following table lists available scripts:

Table 1: Available scripts

SCRIPT	STEP
host_preseparate.py	This script is called on the command station during the compilation before separating the model.
host_postgenerate.py	This script is called on the command station during the compilation after the model's code generation.
target_precompile.py	This script is called on the target during the compilation before compiling the model.
host_postcompile.py	This script is called on the command station at the end of the compilation.
host_reload.py	This script is called on the command station during the execution before loading the model.
target_reload.py	This script is called on the target during the execution before executing the model.
target_postreset.py	This script is called on the target during the execution after resetting the model.
host_postreset.py	This script is called on the command station during the execution after resetting the model. This script could be used to do some analysis after the model execution.

Using local user scripts

In order to execute a local user script (only for this model), just create the corresponding file to the step you want to customize in the model's directory on the command station. The file will be automatically transferred on target if required and executed. Many user scripts for different steps could be created. In order to disable a script, rename the file to a different name.

Using global user scripts

In order to share a script among models on the command station or target(s), the same python scripts are located under `RTLAB_ROOT\common\python\rtlab\global`. Modify these files in order to create a global execution of your script. Remember that these scripts will be applied to every models, they have to be modified with caution.

Debugging

Installing debugger on a target

QNX 6.x

Refer to the RT-LAB Installation Guide to install DDD packages.

Once the DDD debugger has been installed, verify that it is running correctly:

1. Start Photon on the target (if it is not already done) and log in as root.
2. Open a terminal.
3. Enter "ddd" command. DDD should be displayed on the screen.

Red Hat

The DDD debugger has been installed with the operating system. No other installation is necessary.

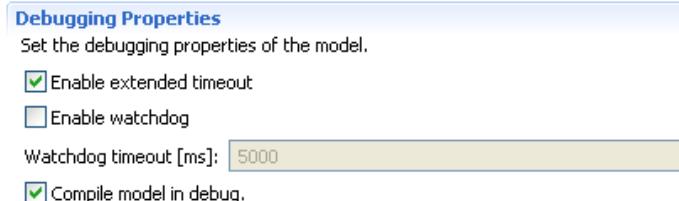
Install the Nightstar Suite from Concurrent Computer Corporation if you want to debug your model using NightView.

Windows

The Microsoft Visual Studio 6 debugger will be used to debug the model. Since MS Visual studio is required in order to compile a model on Windows target, no more steps are required to install the debugger.

Setting RT-LAB configuration

1. Select the model you want to debug in your project.
2. In **Diagnostic Page** from **Model Editor**, check "Use extended timeout" and "Compile Model in debug". Uncheck the "Enable Watchdog" checkbox.



3. Build the model.
4. Configure where to display the debugger, in **Environment Variables Page** from **Model Editor** defines (Add) the following variables:

- **QNX 6.x target:**

- LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:/opt/X11R6/lib:/usr/X11R6/lib
- DISPLAY=127.0.0.1:0

- **Red Hat target:**

- DISPLAY=:0.0

Environment Variables

Set the environment variables of your model.

Name	Value	Description	Add...
DISPLAY	127.0.0.1:0		Select...
LD_LIBRARY_PATH	\$LD_LIBRARY_PATH:/opt/X11R6/lib:/usr/X11R6/lib		Edit...
			Delete
			Unset

5. In **Assignation Page** from **Model Editor** check the subsystem(s) you want to debug. (check the Advanced field to add the Debug column)

Assignations

Set the properties of the subsystems and assign them to physical nodes.

Subsystems

Select subsystems to edit their properties:

Name	Assigned node	Platform	XHP	CPU	Debug
sm_master	q197	QNX 6.x	<input type="checkbox"/> OFF	Auto	<input checked="" type="checkbox"/> ON
ss_slave	q197	QNX 6.x	<input type="checkbox"/> OFF	Auto	<input type="checkbox"/> OFF

Debugging the model**Debugging locally on target****QNX 6.x**

To debug a model:

1. Start Photon on the target (if it is not already done) and log as root.
2. Load the model.
3. Open the source file you want to debug and insert your breakpoint if any. for instance (DDD), if you want to debug the mdlStart (Simulink) of "Rtdemo1" model, go to menu **File->Open source...** then select **rtdemo1_1_sm_computation.c** and click on **Open** button. The source file will open. Look for the MdlStart function, put the cursor on the first line and add a breakpoint (click on the Break button on the toolbar). A breakpoint should have been inserted. When the model will be executed, the execution will stop at this breakpoint.
4. DDD debugger should open on the target screen terminal. Go to menu Program->Run again.
5. Go to menu Program->Continue.

Red Hat

By default, the NightView debugger will be used to debug the model. Refer to NightStart documentation to learn how to use it.

If you want to use DDD instead, edit the ini file <RT-LAB_ROOT_DIR>/common/bin/Hardware.config and in the section [General_REDHAWK] ([General_REDHAWK64] for RedHawk 64) and add the following key:

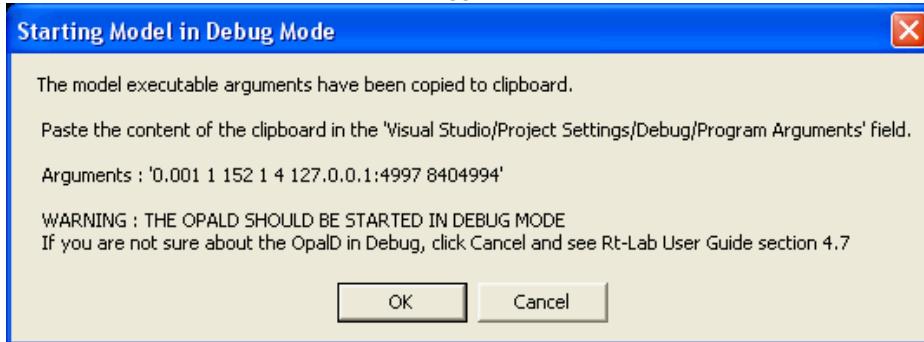
[General_REDHAWK]
DEBUGGER=ddd

To debug a model:

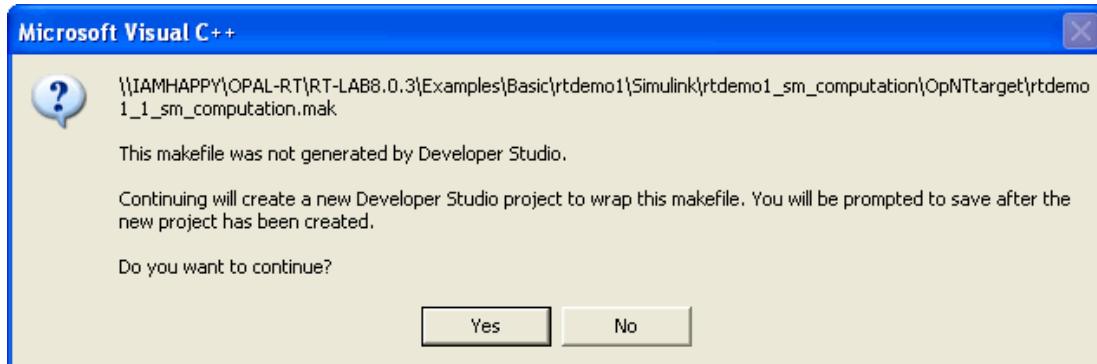
1. Start the graphical interface on the target (if it is not already done) and log as root.
2. Load the model.
3. Open the source file you want to debug and insert your breakpoint if any. for instance (DDD), if you want to debug the mdlStart (Simulink) of "Rtdemo1" model, go to menu **File->Open source...** then select **rtdemo1_1_sm_computation.c** and click on **Open** button. The source file will open. Look for the MdlStart function, put the cursor on the first line and add a breakpoint (click on the Break button on the toolbar). A breakpoint should have been inserted. When the model will be executed, the execution will stop at this breakpoint.
4. Debugger should open on the target screen terminal. (DDD) Go to menu **Program->Run again.**

Windows

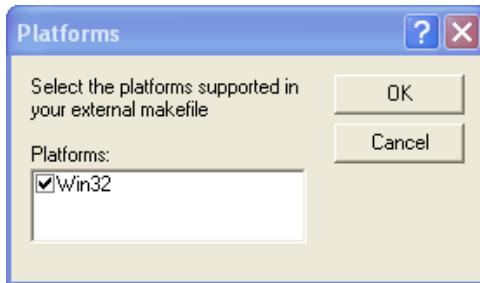
1. Load the model
2. Model argument are copied to the clipboard. These arguments will be used later in order to start the debugger. Press OK.



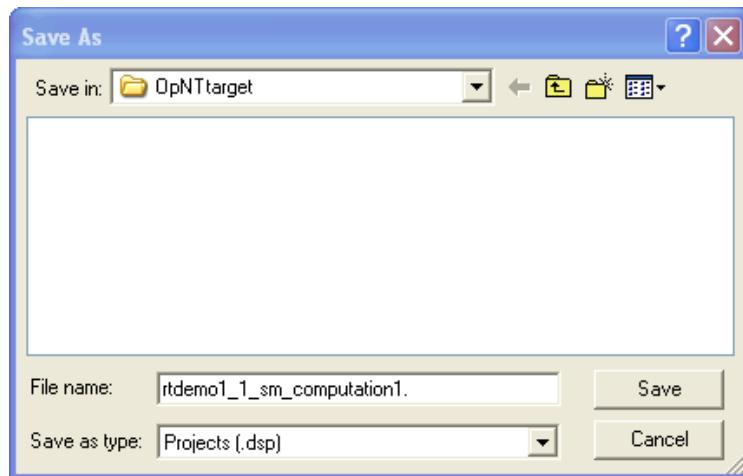
3. MS Visual Studio open and ask to create a projet. Press Yes.



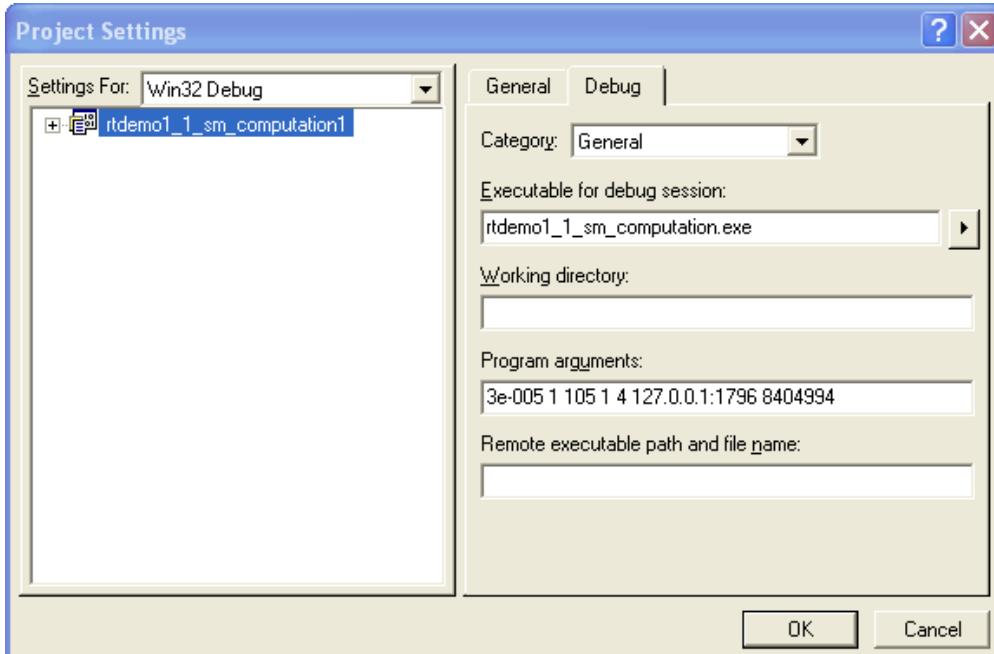
4. At the next screen, press OK.



5. At the next screen, press Save.



6. Go to menu Project->settings



7. Open source file you want to debug and insert breakpoints. Press F5 to start the debugger.

Debugging remotely from the command station (Windows)

QNX 6.x or RedHat targets

In order to debug remotely, an application called “X server” is required to display the debugger. This manual explain how to use cygwin (one of the most popular) as an X server.

1. Download and install Cygwin at <http://www.cygwin.com> (make sure that the Xfree86 package is selected)
2. Start Cygwin
3. Start the X server: **XWin -screen 0 1024 768 -emulate3buttons &**
4. Set the DISPLAY: export DISPLAY=<IP_ADDRESS_OF_WINDOWS>:0.0
5. Start the window manager: **wmaker &**
6. Enable remote access: **xhost +**
7. On paragraph **Setting RT-LAB configuration on page 472**, replace DISPLAY value by DISPLAY=<IP_ADDRESS_OF_WINDOWS>:0.0 (for instance DISPLAY=192.168.0.1:0.0).
8. Load model. DDD should open in the X server.

Windows targets

This option is not supported with this target.

Acquiring and Viewing Data

This chapter discusses aspects related to the acquisition and viewing of your simulation data, including acquisition groups and their parameters.

Acquisition Groups

RT-LAB enables signal acquisition from the real-time target nodes in the cluster. Because TCP/IP is relatively slow compared with most real-time systems, RT-LAB enables you to set different groups of acquisition. Each group can include many signals, including the number of signals per frame to receive, decimation factor, etc. These parameters can be defined separately for each signal. For example, in a system with a 1 millisecond step, a user might want to acquire the data values of a slow changing signal once every 100 steps or so. Signals can also be acquired only once specific conditions have been met, by defining the conditions' parameters in a Trigger block, as described below.

Signals received on the Console (SC_) subsystem are grouped into twenty-four groups. The acquisition group for a signal is specified in the OpComm communication block through which the signal passes.

Acquisition Parameters

Acquisition parameters are shared by all signals included in a given acquisition group. These parameters are specified by you in the Probe Control Panel, on the tabs numbered with the appropriate acquisition group number.

Use **Tools > Probe Control** or click on the Probe Control Icon  to open this panel.



For more details on how to control data acquisition parameters, see .

Triggering Acquisition

By default, each acquisition group is always triggered, meaning that a new data packet is filled as soon as the previous packet is sent. Sometimes, however, you may want to receive data only when a certain condition occurs.

For this reason, the OpTrigger block can be inserted into the subsystem containing the acquisition group signals. The OpTrigger block is used to acquire data triggered by a specific signal, enabling you to acquire information from a specified parts of the simulation only. For complete details about OpTrigger and its parameters, see the online help.

Recording Data with OpWriteFile

To obtain all data for any particular signal, use the OpWriteFile block. Data recorded with this block is stored on the target real-time node's hard drive. The data file is automatically transferred to the command station when the model is reset.

Sometimes, in simulations with fast sampling times, the acquisition data loss on the command station is too significant to clearly understand what is happening in the real-time model. The Simulink library **ToFile** block cannot be used in real-time simulations because it does not have a buffering system.

Using this block in a real-time context without a buffer would significantly increase the real-time simulation's effective step size.

A real-time write-to-file block (**OpWriteFile**) can be inserted into the computation subsystem. This block enables data in MATFILE format to be saved to the real-time node's hard disk. The generated MATFILE (.mat file) can be transferred to the command station and analyzed using MATLAB. The write-to file does not affect the simulation step size in synchronized mode, because all writing is done during the simulation's spare time. The OpWriteFile block uses a circular buffer architecture that enables continuous data acquisition. If the circular buffer is full, it will be emptied completely before starting again. For simulations with small step size, a large buffer size is recommended to avoid discontinuous data in the write-to file.

The Simulink block has one input: the signals to be saved. You can save multiple signals by using a vector as the block input. Each **OpWriteFile** block must be associated with an acquisition group number.

Moreover, because the write-to-file block is considered an extra acquisition group, an **OpTrigger** block is used to write data when certain conditions occurs.

For additional information on **OpWriteFile** and its parameters consult online Help.

Understanding Data Reception

Communication Between the Console and the Target Nodes

The **Console** is connected to each computation node by an Ethernet network. Because this type of network is much slower than a FireWire-type network and because a command station is very resource-demanding, the command station may probably not be able to receive and display all of the data, so there is a need to be selective with respect to incoming data.

Often, data is read only every 2, 3, or even 5 calculation steps: this lightens the load imposed on the **Console**. To accelerate the transmission process, target nodes send batches of data to the network rather than sending one batch per data item.

The target node contains a buffer where it gathers data from many signals for transmission to the **Console**. The target node dispatches its data only when this buffer reaches the level specified in the **Probe Control** panel.

A frame is composed of all of the data that represents a specific signal inside the buffer. For example, if you have three signals and want the data transmitted at every 5000th value, the buffer contains 15,000 samples, or three frames of 5000 values each. Because the command station takes a certain amount of time to receive and display the acquired values and because the real-time model is calculated in continuous mode, it is possible that the next data frame received may not immediately follow the currently displayed frame; in other words, there may be "holes" in the data display. The following figures illustrate this point:

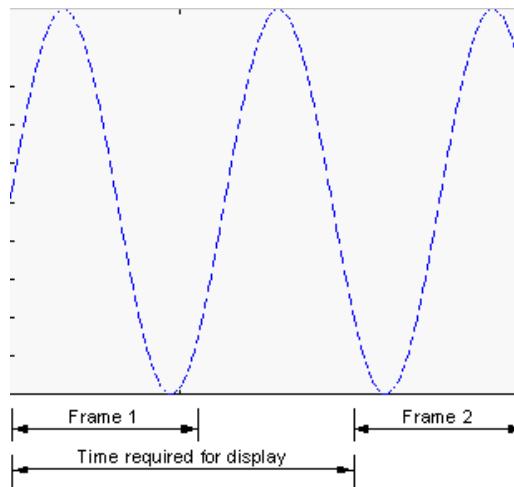


Figure 7:Expected Signal in Data Display

As the time required to display Frame 1 is longer than the frame itself, the portion of the calculation that is performed between dispatching the two frames is lost, creating a discontinuity in the display, as shown below:

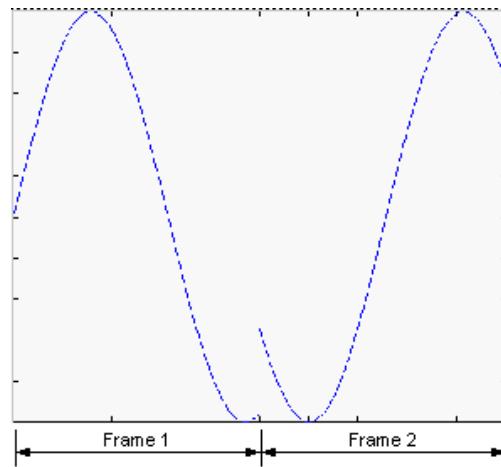


Figure 8:Hole in Data Display

It is possible to minimize this effect. For a number of values per given signal, an increase in the decimation factor (from one acquisition per step to one acquisition for every two steps) also increases the effective length of the frame, diminishing the gap in values between the two data transmissions:

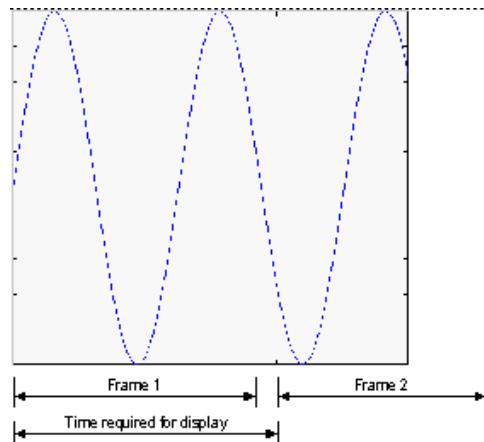


Figure 9:Increasing Effective Length of Frame

Although the display is more representative of the real-time calculation performed, an increase in the decimation factor can cause problems with display resolution, as shown here:

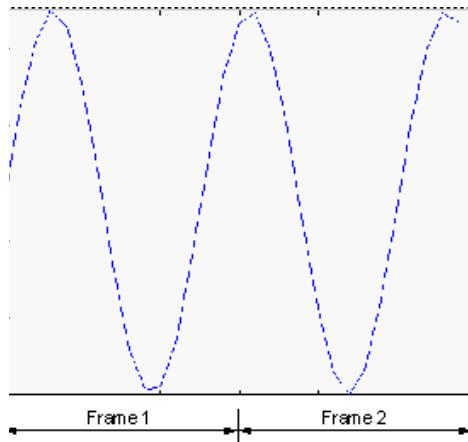


Figure 10:Display Resolution Error

In RT-LAB, even with a command station crash, a real-time model running on the target nodes continues to run. After rebooting the command station, reconnect to the real-time model and continue data acquisition.

Synchronous Acquisition

Remember, acquisition frames are sent when the system has time; frames sent as TCP/IP packets are sent while the model is waiting for a communication packet or for the synchronization pulse. This results in your real-time system not being disturbed by a TCP/IP transmission.

Therefore, the console display could miss some data if the command station is too slow or if the target node does not have enough time left. This technique helps ensure hard-real-time performance.

Aliasing is another acquisition issue which arises with the use of small frames. As described previously, it is possible that the next data frame received may not immediately follow the frame currently displayed. Consequently, in case of small frames (like one value per frame) the possibility of losing some values results in incorrect data display. To illustrate this concept, let's assume that we have a sine signal as shown here:

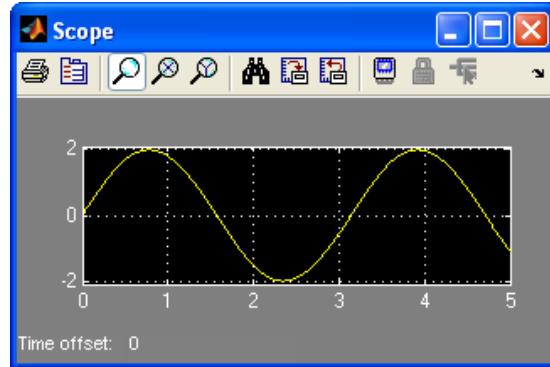


Figure 11:Figure with no Aliasing

If, for example, data is acquired once every n steps because acquisition cannot follow the real-time simulation, the display shows values without properly updating the time axis, which results in aliasing: the frequency of the resulting signal is increased by a factor of n .

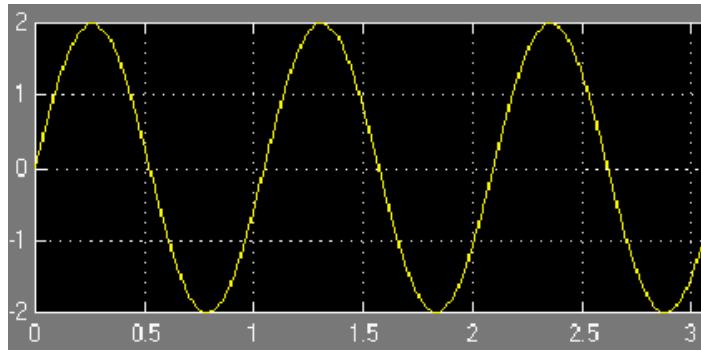


Figure 12:Aliasing Non-synchronized Time Axis

The data reception problem occurs because the **Console** isn't aware that data from the target node is being missed. To circumvent this problem, it is possible to synchronize the **Console** time with the simulation time to avoid data loss and data reception errors.

During acquisition, the console subsystem acquires signals from the simulation platform, enabling you to view data (Scope, To Workspace-type blocks, etc.). There are three options to choose from with the **OpComm** block mask.

Table 2: OpComm block Mask Option

BUTTON / FIELD	FUNCTION
Enable synchronization	Enables the synchronization algorithm for a given acquisition group. This algorithm must be used if synchronization is required between acquisition Console time and simulation time. In case of a synchronization problem, the Console detects missing data from the target node and the algorithm either replaces the lost data with the last signal value received, or interpolates the data.

Table 2: OpComm block Mask Option

Enable interpolation	In case of missed data from a computation subsystem, the synchronization algorithm interpolates data during a range of data loss. The following illustrations make this point clear. The display at left indicates no interpolation during data acquisition while the display at right indicates interpolation between two frames during data loss.
Threshold	Difference between the simulation Console time and the simulation target time. Whenever this difference is exceeded, the synchronization algorithm stops interpolating and re-synchronizes to the new value.

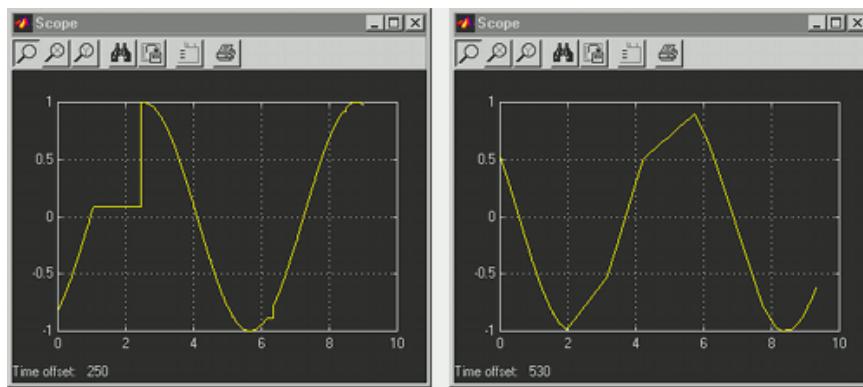


Figure 13:Interpolation in data acquisition

Monitoring Models

This chapter describes how to use RT-LAB to monitor the performance of your running simulation.

Monitoring Overview

The Monitoring is a profiler that is tightly integrated with RT-LAB and runs within your simulation on the target computers. It enables you to profile your simulations for performance, CPU usage, overruns, and synchronization problems. Monitoring provides a comprehensive set of features and provides an intuitive user interface to analyze the simulation behavior.

The monitoring features provides:

CPU profiling

- Find out where your CPU time is going including I/Os calculations and RT-LAB overhead.
- Identify performance bottlenecks.
- Get detailed information on the CPU usage when overruns occur.
- Evaluate how much time is left in your computation step for adding calculations, I/O access, etc.

Tasks profiling

- Check the time spent on your user written code or your block execution in the model.
- Determine the tasks' execution order and the concurrent execution.
- Scale your application by evaluating the effect of distribution.
- Identify and resolve deadlocks.

Distributed monitoring

- The Monitoring give you the ability to concurrently monitor many processes running on many CPU, thereby profiling a distributed simulation.

Monitoring Architecture and Concepts

The Monitoring service is a service running on the target that records the exact time of events that occur during the target simulation. All measurements are relative to the beginning of the simulation and are measured using external hardware IO clock (in hardware synchronized mode) or using processor clock (in software synchronized mode).

The monitoring service also calculates time elapsed between these events. These measurements are called *probes*. Generally the probes are the time elapsed between two events that occur on the same time step, i.e. the beginning and the end of a task. Tasks are any calculations performed on the target during the real-time simulation. Also note that most common tasks are completely executed during one single step and are executed at every step of the simulation.

The monitoring service measures all time spent by all RT-LAB services including acquisition, communication and IOs. It could also measure time spent by other tasks, such as user C Code function called during simulation, or time spent by one or more subsystems in the model's diagram. The most common and useful probes are:

Major and minor computations

Time spent by the model to perform blocks' calculation including algebraic calculation, discrete and continuous states calculation. See [Continuous Sample Time and Major/Minor Time Step](#) for more detail on major and minor computations.

- **Execution cycle**

Time spent to compute all tasks of the model (major and minor computation) and RT-LAB services and overhead (status update and communication, IOs, ...)

- **Total step size**

Time to execute one time step of the model. It is very close to the model's time step and is limited by the clock's precision used to synchronize the simulation.

- **Total idle**

Available time that can be used to compute more calculation.

The monitoring service also performs statistical analysis and summarizes this information to detect where the CPU time is going.

The monitoring service is based on the RT-LAB acquisition system and performs data acquisition similarly to normal acquisition. It means that the monitoring service performs data acquisition for N consecutive steps. Data acquisition can be triggered and acquired data can be saved to a file. See [Acquiring and Viewing Data](#) for more information on the acquisition concepts and features.

Acquired probes are sent to the host computer and can be displayed in the [Monitoring View](#).

Enabling Monitoring

In order to be able to monitor a model, the monitoring service must be enabled before loading the model on target. Since the monitoring will add a small overhead to your model's calculation time (a few microseconds depending on the target CPU speed), it is not enabled by default.

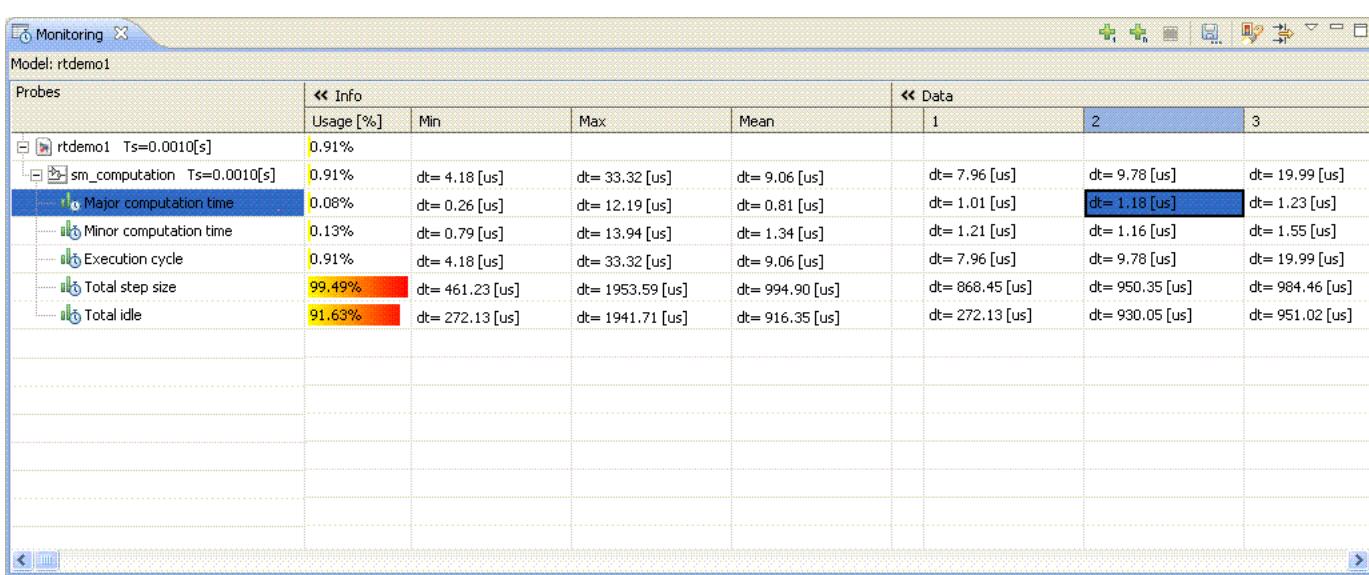
To enable the monitoring:

1. Select the model in the Project Explorer view.
2. Double click on the model, to open its editor.
3. Select the diagnostic tab. (refer to [Diagnostic Page](#) for more information.)
4. Check the "Enable monitoring" property.

Note: if the OpMonitor block is inserted into the model, the monitoring is automatically enabled when the model is compiled and loaded on target.

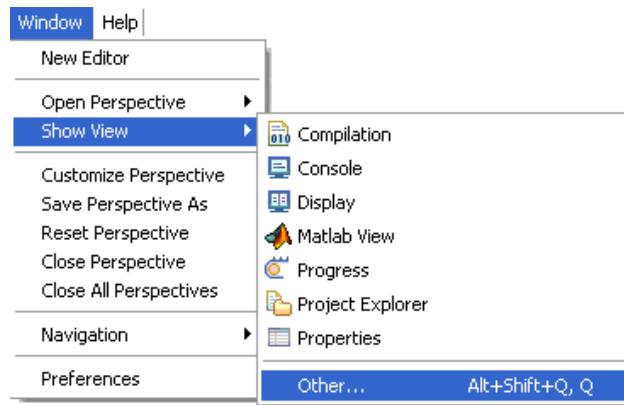
Using the Monitoring View

The [Monitoring View](#) displays CPU usage of each task/probe executed during real-time simulation. It displays the start and stop time of each probe and its duration. It detects overruns and highlights steps where overruns are detected. Here is an example of its contents:

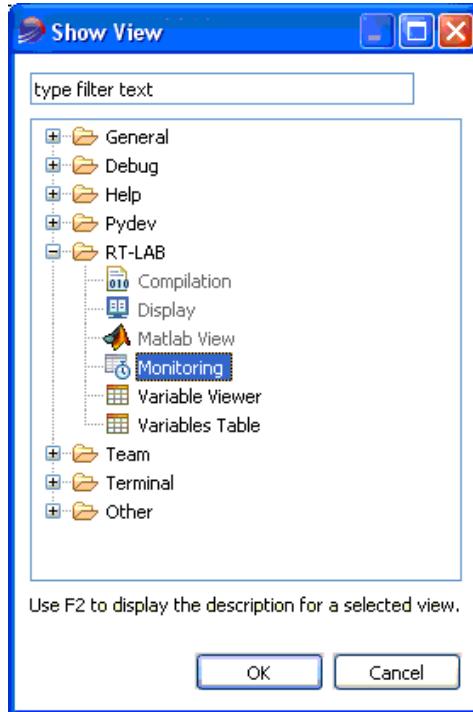


The **Monitoring View** can be used with any model that has the monitoring setting enabled. You do not have to rebuild your model once you have enabled monitoring. Once the model has been selected in the Project Explorer and loaded on a target, simply open the **Monitoring View** to get the monitoring information:

- Click **Window > Show View > Other ...** menu,



- and then select the **RT-LAB > Monitoring** view.



- Click the **Get Probes** button in the view's toolbar to retrieve monitoring information.

Refer to [Monitoring View](#) for more information.

Using the OpMonitor Block

The OpMonitor block provides monitoring and probe information inside the Simulink model as outputs of the block. Most common probes could be outputed from the blocks.

The model can use these values to adjust itself by enabling or disabling subsystems or to log the information to a data logging file, etc. When the block is inserted into the model, the monitoring will be enabled the next time the model is compiled and then loaded. In this case, you do not need to enable the monitoring feature using the [Diagnostic Page](#) of the model editor. Refer to the *Block Library Reference Guide* for more information about the **OpMonitor** block.

Measuring a Model's Subsystem Calculation Time

RT-LAB monitoring provides information about RT-LAB overhead and calculation time of the whole model. Sometimes, only a part of the Simulink model needs to be monitored. In this case, user probes could be added to the model's diagram using the OpMonitoringStart and OpMonitoringStop blocks. These blocks allow you to measure the time spent by one subsystem on the model that is connected to these blocks. Refer to the *Block Library Reference Guide* for more information how to insert these blocks into the model.

Measuring User Code Source Calculation Time

User C code could be added to the Simulink model using S-functions or UCBs. This user code is part of the major and minor calculation time event. At times, parts of the user code need to be monitor. This can be achieved using the monitoring API functions provided by RT-LAB.

- `OpalMonitoring_Register`: Registers a new event name and type.
- `OpalMonitoring_LogStart`: Starts logging the timing of the specified event.

-
- OpalMonitoring_LogStop: Stops logging the timing of the specified event.

See the Monitoring1 template to learn how to insert monitoring probes within user code.

- Create a new RT-LAB Project.
- Type its name and then click **Next**.
- Select the Features/Monitoring1 template from the list of available project templates.
- Click **Finish**.

Events and Probes List

Here is the list of all available probes and events that can occur when a simulation is running:

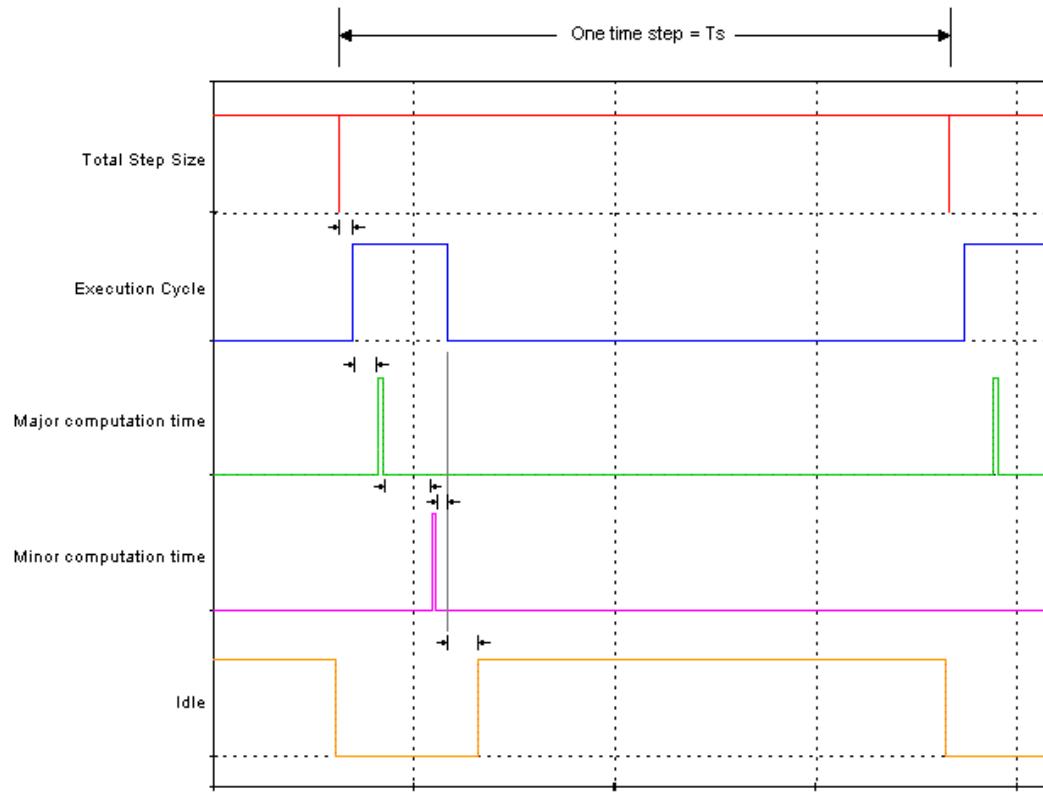
EVENT/PROBE NAME	DESCRIPTION	FREQUENCY
Basic		
Major computation time	Time to compute the model's calculation at every major step. It depends on the model's solver used to compute the results. The major step includes a part of the algebraic calculation, the discrete states calculation and the major continuous states calculation. See below for more detail.	Every step
Minor computation time	Time to compute the model's calculation at every minor step. It depends on the model's solver used to compute the results. The minor step includes a part of the algebraic calculation and the minor continuous states calculation. See below for more detail.	Every step
Execution Cycle	Time to compute model's computation time (major and minor) + RT-LAB overhead for all services (status update and communication, IOs).	Every step
Total Step Size	Execution Cycle + time to wait the clock's interrupt. Normally it should be very close to the model step size.	Every step
Idle	Available time that can be used to compute more calculations.	Every step
Advanced		
Sync Handler Jitter	Switching time between the Interrupt handler and the main process of the model. Only in software or hardware synchronized mode. It is used to determine when the step will begin.	Every step
Synchronized IOs	Time to acquire data from the IOs board at the beginning of the step.	Every step
Status Update	Time to update internal status of the subsystem.	Every step
Data Acquisition	Time to acquire model's data. It does not take into account the time to transfer the data to the Command Station.	Every step
Signals Change	Time to update model's signals.	Only when signals are received from the Console
User Blobs	Time to receive and call the user callback function registered for a specific blob.	Only when a blob is received

EVENT/PROBE NAME	DESCRIPTION	FREQUENCY
Send Status	Time to send the internal status to other computation node. The Time is 0 if the model is not distributed	Every step
Multi-Receive	Time to receive status and data from other computation nodes. The Time is 0 if the model is not distributed.	Every step
Handle target requests	Time to handle feedback sent to the Host Command Station	Only upon request
Handle host requests	Time to handle the commands received from the Host Command Station	Only upon request
Pre-Execution computation time	Time to execute "Pre-Execute" callback. This computation is done before executing model blocks. It is generally used by IO blocks.	Every step
Post-Execution computation time	Time to execute "Post-Execute" callback. This computation is done after executing model blocks. It is generally used by IO blocks.	Every step
Number of Overruns	Number of overruns. An overrun occurs when the clock's interrupt is raised before the end of the previous calculation step. It indicates the number of overruns recorded since the simulation began, and only for clocked subsystems.	When Idle is negative

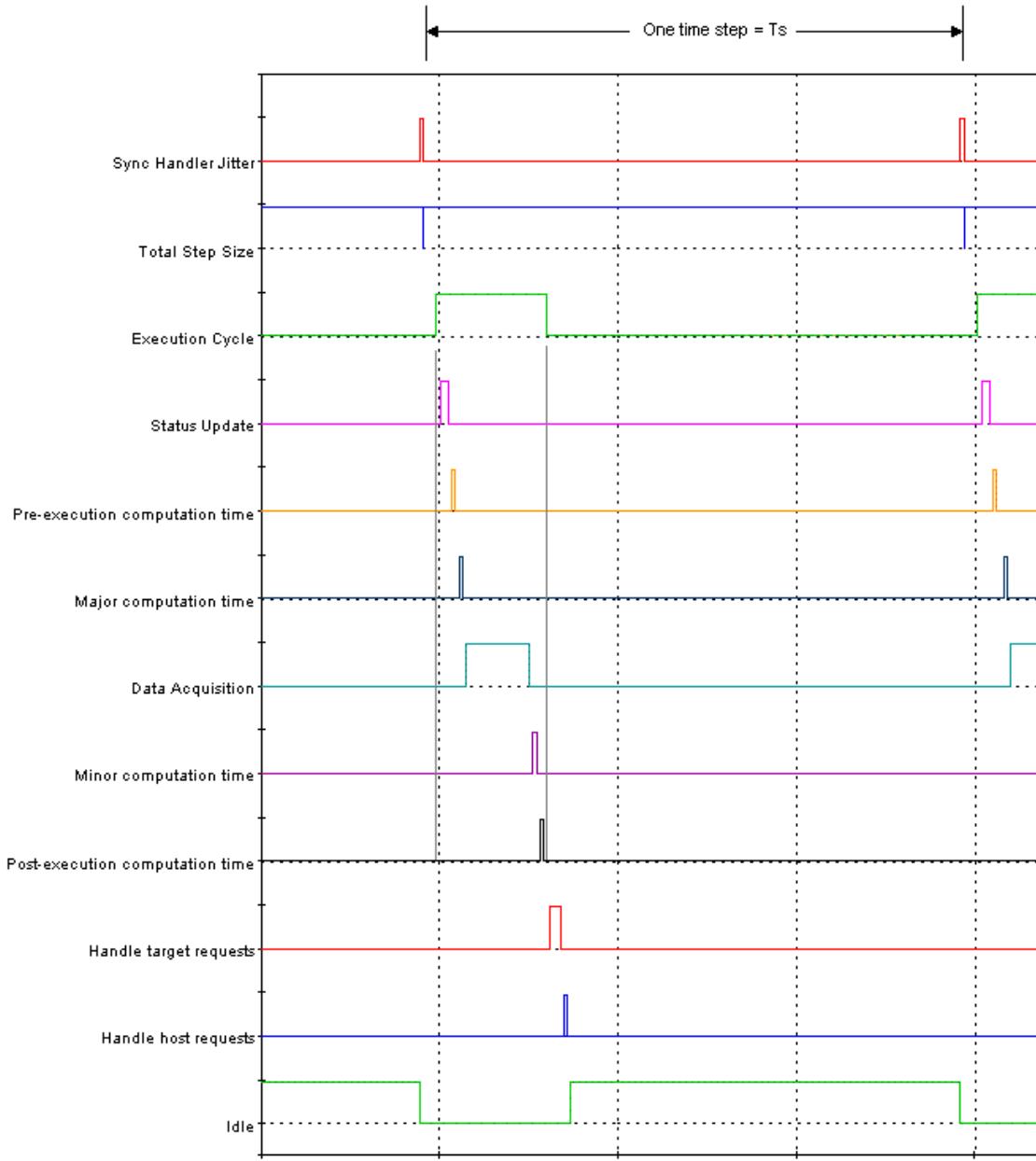
The following figure shows an example of the basic probes acquired by the monitoring over time. It illustrates the data acquired for one time step of simulation for a model with one subsystem. Here is a short description of the probes shown on this graph.

- The total step size shows the time to simulate one time step in real time; it includes all other probes.
- The execution cycle tasks is the time spent by model to perform all model/blocks calculations and the RT-LAB services.
- The major and minor computations tasks is the time spent by the model to perform only all blocks calculations.
- The idle task is the time available that can be used to compute more calculation.

Pay attention: The major and minor calculation tasks are concurrent to the execution cycle. Also note that small holes are visible between tasks. These holes represent the time spent by other tasks not shown on this figure.



The following figure shows a similar example but now it also includes most advanced probes. It illustrates the data acquired for one time step of simulation for a model with one subsystem. Pay attention to the order the tasks are executed. You can also see that most of the CPU time is now represented by one of the tasks (there is no hole between tasks).



Continuous Sample Time and Major/Minor Time Step

Unlike the discrete sample time, continuous sample times are divided into major time steps and minor time steps, where the minor steps represent subdivisions of the major steps. The discrete ODE solver you choose integrates all continuous states from the simulation start time to a given major or minor time step. The solver determines the times of the minor steps and uses the results at the minor time steps to improve the accuracy of the results at the major time steps. However, you see the block output only at the major time steps.

The number of subdivisions performed by the discrete solver depends on the ODE solver. For example, the ODE3 and the ODE5 solver perform 3 and 5 iterations, respectively.

Using Python Script and the Macro Recorder

This chapter describes how to use Python script and the macro recorder.

Macro Recorder Overview

The macro recorder is a utility that automates repetitive and tedious tasks done on the real-time simulator. It records all actions performed with RT-LAB and saves them as a set of API commands in the Python native language (see the *Python API Reference Guide* for more details on the RT-LAB API). The output of the macro recorder is a Python script module. Once the macro is recorded, RT-LAB lets you modify and edit the macro (script) using a rich editor and then easily play it back to perform simple or complex sets of tests without needing programming skills.

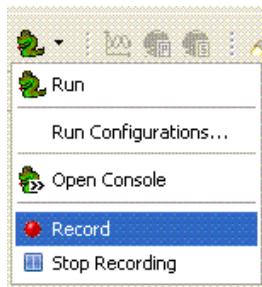
The macro recorder is often used while changing model parameters and interacting with the behaviour of the simulation. The recorded macro can be used to create sequences of manoeuvres to control the simulator and model. It is also possible to manage the simulator's execution mode and configure all simulator parameters and settings.

Before you start to record a new macro, plan the actions you want to perform on the project and simulator. Errors and corrections are also recorded. RT-LAB stores the result in a Python script function.

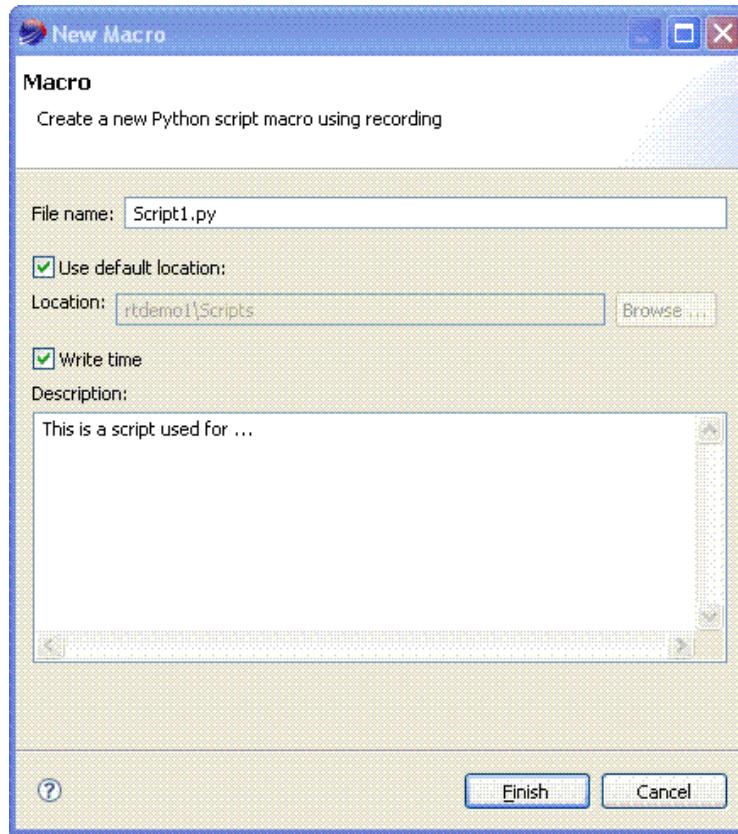
Recording a New Macro

Follow these steps to record a new macro:

- Select the project from the **Project Explorer** view; only actions performed on this project will be recorded.
- On the **Tools > Python** menu, click **Record** item, or simply click the **Record** item in the drop down menu of the **Python** toolbar button.



The **New Macro** dialog box will appear:



- In the **New Macro** dialog, type the script's file name where it will be stored. By default, the script will be located in the project's scripts folder, but it is also possible to specify a different location such as a project's folder or anywhere on file system.
- Check/uncheck the **Write time** setting to record the time elapsed between actions in the macro. When checked, the macro recorder will add a sleep command between all actions to simulate the time elapsed when recording.
- Type the description for your macro.
- Click **Finish** to start recording.
- Perform actions on your project, such as changing parameters or configurations. These actions will be recorded.
- On the **Macro Recorder** toolbar, click the pause and resume buttons to activate and deactivate recording project actions.



- To stop the recording, click one of the following item:
 - Stop recording** button from the **Macro Recorder** toolbar,
 - Stop recording** item menu in the **Tools > Python** menu or
 - Stop recording** item menu in the Python drop down menu in the main toolbar.
- RT-LAB will automatically open the Python editor for the newly recorded macro.

-
- To cancel the recording, click the **Cancel recording** item in the **Macro recorder** toolbar.

Opening a Macro/Script

Follow these steps to open an existing macro/script:

- Click **Open file** from the file menu and locate your file by browsing your disk or
- Double click the Python script macro in the **Project Explorer**.
 - Make sure that your project's file system is visible by clicking the **Filter resources** button in the **Project Explorer** view.
 - Expand your project in the Project Explorer view and locate the Python script in the file system.
 - Double click the Python script macro.
- Drag and drop a Python script macro from the Windows Explorer to the RT-LAB editor area.

Editing a Macro/Script

The macro/script editor has a rich set of functionalities that let you edit your macro/script easily. Here is a brief description of the available features:

- Syntax highlighting : Provide different colors and text styles to be given to dozens of different lexical sub-elements of the Python syntax. These include keywords, comments, control-flow statements, variables, and other elements. Programmers often heavily customize their settings in an attempt to show as much useful information as possible without making the code difficult to read.
- Code analysis : Provides error finding in python programs. It finds common errors such as undefined tokens, duplicated signatures and warns about things such as unused variables or unused imports.
- Code completion: provides context-sensitive completions and is enabled with **Ctrl+Space** key in the editor.
- Refactoring : Provides functionalities to change the source code to improve code readability, reduce complexity and improve the maintainability of the source code such as renaming, extracting, inlining code elements.
- Navigation Views: Outline View, Hierarchy View, ...
- Content Assistants: Helps user to write code faster and more efficiently by providing list of accessible keywords such as variable, method, import, module, ...

Once your script is open in the editor, perform the following steps to edit your script:

- Locate the part of your script you want to edit.
- Type your text.
- Save your file by clicking the **Save** button in the main toolbar.

Running a Macro/Script

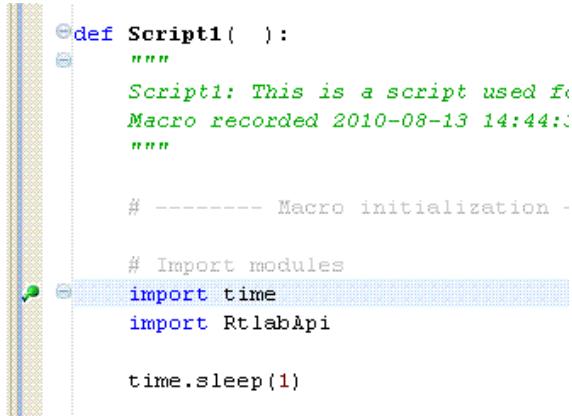
Follow these steps to run a macro:

- Select the Python script from the **Project Explorer** or click inside your script in the editor area.
- On the **Tools > Python** menu, click **Run** item (**Ctrl-F11**), or simply click the **Run** item in the drop down menu of the **Python** toolbar.

Debugging a Macro/Script

Follow these steps to debug a macro:

- Open the Python script macro. See [Opening a Macro/Script](#) for more information.
- Add break point to your script:
 - Double-click the script's left ruler or
 - Type Ctrl+F10 to open the context-menu. If everything goes ok, you'll have the breakpoint shown in your sidebar (as below).



```

def Script1( ):
    """
    Script1: This is a script used for
    Macro recorded 2010-08-13 14:44:00
    """

    # ----- Macro initialization -

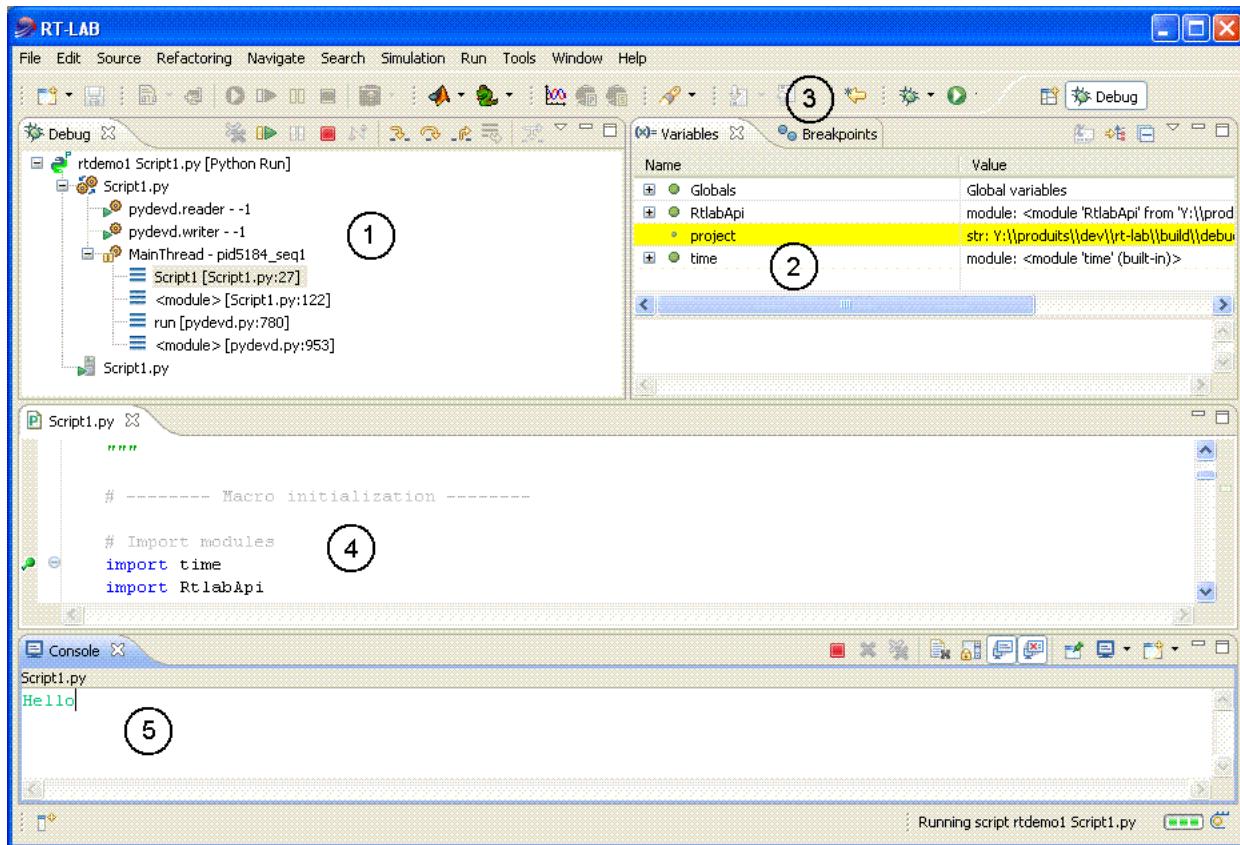
    # Import modules
    import time
    import RtlabApi

    time.sleep(1)
  
```

- Right click the file and choose to debug the file as a **python run**. NOTE: if you want to re-run the last executed file, you can click **F11** to debug it.

When it hits the breakpoint, it will ask you to go to the debug perspective (you should click **YES** to this dialog).

This perspective has the actions needed for debugging and allows you to see the variables and evaluate expressions.



1. Stack view: You can see the variables in previous stacks by clicking on the stack you want to view.
2. Variables view: Allows you to see the globals and locals for the selected stack.
3. Breakpoints view: Selecting this tab allows you to see the breakpoints available and enable/disable any breakpoint.
4. Editor: Shows the code and highlights the line that is about to be executed.
5. Console: Shows the output of the Python script macro.

Now, the basic things you can do in the debugger have some keybindings:

- Step in: F5
- Step over: F6
- Step out: F7
- Resume: F8

Using RT-LAB Blocks

RT-LAB comes with a library of blocks that can be added to your model. These blocks enable you to access various types of I/O modules and other functions.

I/O and synchronization module blocks serve to control external equipment; generic RT-LAB blocks provide other network communication and timing functions.

This chapter discusses the need, use and parameters of the I/O blocks, synchronization blocks and other generic blocks that can be used with RT-LAB.

RT-LAB block libraries can be found in the Simulink library browser.

Generic blocks

RT-LAB comes with many blocks providing generic functionalities like data logging, monitoring, communication with external composants. These blocks need to be inserted in the RT-LAB subsystems (SM, SS or SC). Some blocks work offline, others work only when in real-time simulation. Refer to the online documentation in order to get a detailed description for each block.

Adding Blocks to Model

To add blocks to your model and set their parameters:

1. Open the browser and click on the RT-LAB block that corresponds to the desired function.
2. Drag and drop the RT-LAB block into your simulation model.
3. Double-click the RT-LAB block to change its parameters. This opens a parameter control box.

I/O blocks

I/O cards enable you to control external equipment. These cards may be digital to analog (D/A) converters, analog to digital (A/D) converters, input/output (I/O) binary ports and quadrature decoders, among others. Synchronization cards synchronize computation and communication between nodes.

RT-LAB comes with a library of blocks that provide access to different types of I/O card. The library of modules are sorted by vendor name, product name then block name. The RT-LAB I/O library can be found next to the RT-LAB library in the Simulink library browser.

Each block refers to a function provided by the I/O board. For each block inserted into a simulation model, different parameters must be set to indicate to RT-LAB where to find the card or module that corresponds to the block.

These blocks have no effect when running offline. When loading the model, the target subsystem where the I/O block has been inserted (**SM_** or **SS_**) must be assigned to the target node with the corresponding I/O card.

Refer to the online documentation to get more information about each block.

RT-LAB Connectivity

This chapter describes RT-LAB add-ons that expand the capabilities of RT-LAB :

- **RT-LAB Orchestra,**
- **RT-LAB Asynchronous processes,**
- **RT-LAB User SFunction**
- **RT-LAB / Xilinx System Generator toolbox integration.**

RT-LAB Orchestra

Introduction

RT-LAB Orchestra is an add-on that extends RT-LAB's connectivity capabilities to heterogeneous co-simulations. Heterogeneous co-simulations consist of simulations written in different programming languages, or generated by various simulation tools. Although Simulink provides some connectivity capabilities through the use of S-functions, the main advantage of using RT-LAB Orchestra is flexibility: co-simulation components can be developed and tested by different teams, then integrated to form a cohesive co-simulation system.

When using RT-LAB Orchestra, RT-LAB is referred to as the Framework, and simulation components that exchange data with this Framework are referred to as External Components. RT-LAB Orchestra provides a real-time, user-configurable communication layer that sits on top of the RT-LAB Framework. In this context, RT-LAB becomes a backbone that provides a transport layer between simulation components. Currently supported External Components are:

- C code applications,
- standard (non RT-LAB and RT-LAB) Simulink models,
- standalone Dymola models.

As a software application, RT-LAB Orchestra consists of the following entities:

- a configuration file used to specify the simulation data exchange between the Framework and External Components; the communication layer depends on the content of this configuration file,
- a library of Simulink blocks, the rrlab_orchestra Simulink library, that provides interfaces between the Framework and External Components,
- a C code application programming interface, the Orchestra RT-API, which is used by External Components to exchange simulation data with the Framework. Calls to the RT-API are embedded in an External Component where the data exchange takes place.

Each of these entities is further described below.

Configuring the RT-LAB Orchestra communication layer

The RT-LAB Orchestra Data Description File

You configure the RT-LAB Orchestra communication layer by specifying the simulation data to be exchanged in an eXtended Markup Language (XML) file called a Data Description File (DDF). A DDF is decomposed into a 4-level hierarchy. Each level is further described in subsequent sections. The figure below shows the structure of a sample DDF.

```

<?xml version="1.0" standalone="no" ?>
- <orchestra>
  - <domain name="uav1">
    <synchronous>yes</synchronous>
    <multiplePublishAllowed>no</multiplePublishAllowed>
    <states>no</states>
    <connection type="local" />
  - <set name="PUBLISH">
    - <item name="Autopilot">
      <type>double</type>
    </item>
    - <item name="Throttle">
      <type>int</type>
      <length>4</length>
    </item>
  </set>
  - <set name="SUBSCRIBE">
    - <item name="Elevation">
      <type>double</type>
      <default>+12.3e4</default>
    </item>
    - <item name="Heading">
      <type>double</type>
      <description>Where the aircraft nose is pointing</description>
      <default>23</default>
    </item>
    - <item name="Roll">
      <type>byte</type>
      <length>1</length>
    </item>
  </set>
</domain>
</orchestra>
```

The Orchestra root element

The `<orchestra>` element is at the top of a DDF. All the XML elements comprised between the opening and closing `<orchestra>` tags are part of the same co-simulation. Syntactically valid XML elements that are not explicitly required to configure RT-LAB Orchestra are ignored. Also note that all XML tags and attributes in a DDF are in lower case.

Domain elements

The next level down from the top element contains a list of `<domain>` elements. An RT-LAB Orchestra domain is a container for named data items exchanged between co-simulation components, also referred to in this context as domain participants. A domain is defined by the `name` attribute of its element. A domain name has to be unique inside the root element.

A `<domain>` element contains several Quality Of Services (QoS) attributes that define the connection policies followed by domain participants:

-
- Synchronicity - this parameter determines whether domain participants exchange simulation data synchronously or asynchronously. In the DDF, this attribute is controlled by the `<synchronous>` element,
 - Writer Access Exclusivity - this parameter determines whether several publishers can write to the same domain. Any number of domain participants can subscribe to a given domain at the same time. In the DDF, this attribute is controlled by the `<multiplePublishAllowed>` element,
 - Writer Access Priority - this parameter determines whether an external component needs to seed the data exchange within a given domain, by publishing to that domain first (before the RT-LAB Framework). In the DDF, this attribute is controlled by the `<hasStates>` element.

Note that some connection policies are not compatible: a connection with states has to be synchronous, and a synchronous connection doesn't allow multiple publishers to the same domain. Incompatible connection policies are treated as errors by RT-LAB Orchestra.

In addition, the type of connection to a domain is specified using the `<connection>` element. Supported types are "local" and "remote". For a local connection, the processes associated with the RT-LAB framework and the External Component are located on the same machine and they communicate through a shared memory segment.

For a remote connection, the processes associated with the RT-LAB framework and the External Component are located on different machines and they communicate through a reflective memory segment.

When specifying a remote connection, it is also necessary to provide information about the type of communication card that is used to connect an External Component to the Framework, as well as the PCI index (a positive integer) corresponding to where the card is inserted. This is done by using `<card>` element and `<pcindex>` elements respectively.

Data sets

Each domain is required to contain both a Publish and a Subscribe set, possibly empty. The Publish set contains the list of data items written by the RT-LAB Framework to the communication layer, and therefore read by external domain participants. The Subscribe set contains the list of data items read from the communication layer by the RT-LAB Framework, and therefore published by an external component.

Data items

Data items are the actual signals exchanged within a co-simulation. Data items are identified by a name attribute in a DDF. Item names have to be unique within a given domain.

Data items are further defined by a type and a length. The type corresponds to native types in the C programming language (such as "double" or "int"). The C-code "char" type is denoted by "byte". The `<length>` element is used to represent data items that consist of arrays of native types. For example, a data item defined with the C-code `int[4]` corresponds to the DDF elements `<type>int</type>` and `<length>4</length>`

For documentation purposes, you can insert an optional `<description>` XML element inside an `<item>` element.

You can also assign a default value to data items in the Subscribe set. Valid formats for default values correspond to acceptable definitions for a C code double (e.g.: 1.2, +1e2, .E-2 etc.). Note that it is an error to assign a default value to a data item that is part of the Publish set of a domain; this is because an Orchestra framework publishes all the data items of a Publish set at every time step, therefore default values would be overwritten before they can be read by a subscriber External Component.

Constraints on element names

The following constraints apply to domain and data item names:

- the names are case-sensitive,
- leading and trailing spaces are stripped, and the resulting stripped names are limited to 64 characters,
- the names have to consist of at least one non-space character.
- the names cannot contain a comma (",") or a pound sign ("#").

The RT-LAB Orchestra CSV File

CSV stands for Comma Separated Values, a file format that is compatible with MS Excel. An Orchestra CSV file consists of a header followed by a body made up of rows of data.

The CSV header

The header consists of 2 distinct and required elements, namely a Domain Descriptor and a Column List.

The Domain Descriptor provides a description of a DDF domain in the format of a pair of attribute/value pairs separated by ":", namely following the format: domain attribute1=value1:attribute2=value2 etc.

The table below shows the list of defined attributes and the corresponding valid values.

Attribute	Value	Description
name	any valid identifier	The domain name.
connection	local/remote	The connection type.
cardName	VMIPCI5565-64M VMIPCI5565-128M	The type of reflective memory card used for a remote connection.
pciIndex	any positive integer	The PCI index that corresponds to the communication card used for a remote connection.
sync	yes/no	The Synchronicity QoS.
state	yes/no	The Writer Access Priority QoS.
multiPub	yes/no	The Writer Access Exclusivity QoS.

The Column List is a list of named columns that RT-LAB Orchestra uses to map the data in the CSV file to the various DDF elements. The Column List should contain the following columns: "domain","frameworkToClient","clientToFramework","type","length","default","description".

The columns can be provided in any order, but if the CSV file contains columns in addition to the Column List, then the entire Column List has to come first in the file. The domain names under the "domain" column must be defined in the domain descriptor.

Note that all data in the CSV header is case-sensitive.

The CSV body

The CSV body contains a series of rows, one per data item. The content of each column of the data row is read and parsed based on the corresponding column of the Column List.

Each domain referred to in the CSV body has to appear in the Domain Descriptor. Data items for a given domain do not have to be contiguous; if a data row does not contain a domain name, the corresponding data item is assumed to belong to the domain that was last defined in the body.

Creating a CSV file

You can create a CSV file by editing a text file and saving it with the .csv extension, or by editing an Excel spreadsheet and saving it with the file type set to CSV.

DDF Configurator

RT-LAB Orchestra includes a graphical tool called the DDF Configurator that you can use instead of editing the XML file directly.

Launching the Orchestra DDF Configurator

The DDF Configurator can be launched as a standalone application or from a block of the RT-LAB Orchestra Simulink library.

Double click the file DDFConfig.jar located in <RT-LAB installation dir>\Common\bin to run the DDF Configurator. DDF Configurator is a Java application that requires a Java Runtime Environment to be installed. Refer to the section "Environment Requirements" for further detail.

Once you have created a DDF, you can use it to configure a block from the RT-LAB Orchestra Simulink library. To launch DDF Configurator from a Simulink block, open the RT-LAB Orchestra library - Click the block and select "Configure".

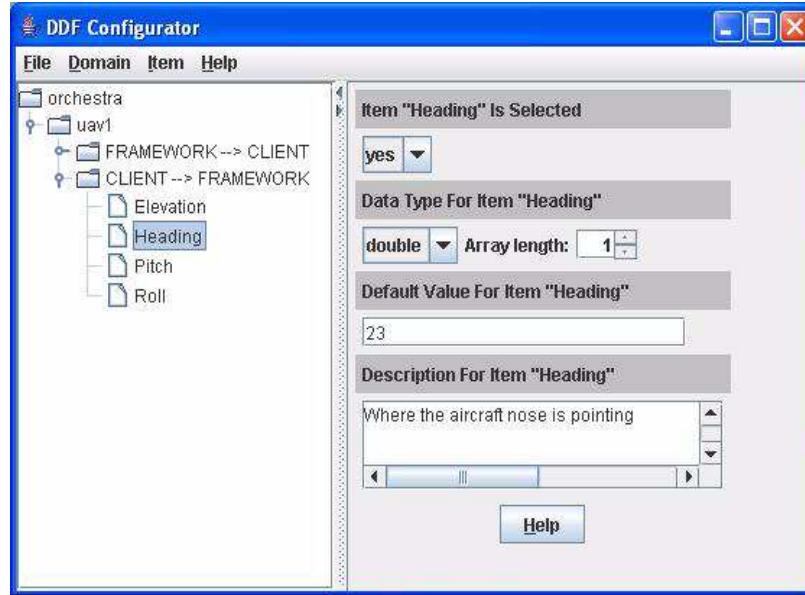
Key features

DDF Configurator allows you to:

- Create a DDF from scratch,
- Import and update an existing DDF,
- Import a Dymola configuration file and create a DDF from it,
- Import data from a CSV file into DDF Configurator,
- Save DDF data to a CSV file.

Graphical views

The DDF Configurator consists of two views separated by a movable divider, and of a toolbar located along the upper edge of the panel. The lefthand side view is called the Explorer View, while the right-hand side view is called the Properties View. The Explorer View is a tree structure that displays the simulation data currently defined under the Orchestra root node. At the top of the Explorer View is the root node. The root node is a container for all the other data elements of a DDF, namely: domains, data sets and item names. You can expand a tree node to view its content. The Properties View displays the properties of the element currently selected in the Explorer View. There are 4 variants of the Properties View that correspond to the 4 level hierarchy of a DDF. The figure below shows a snapshot of the DDF Configurator with the data item "Heading" selected in the Explorer View, and the corresponding Properties View.



Creating a domain

A domain in the Explorer View consists of a Publish set and a Subscribe set . The Publish set, denoted "FRAMEWORK-->CLIENT", contains the data items published (written) by the RT-LAB Framework to the Orchestra communication layer, whereas the Subscribe set, denoted CLIENT-->FRAMEWORK, contains the data subscribed from (read) from the Orchestra communication layer.

There are 2 ways to create a new domain:

- right-click the Orchestra root element in the Explorer View, this causes a pop-up menu to appear. Left-click the "Add Domain" of the pop-up menu, this causes a new domain element to be added to the Explorer View,
- select the "Domain->Add New Domain" item on the toolbar menu.

When created, a domain is automatically assigned the name "new Domain#", where "#" represents a positive integer that increments sequentially. You can then choose to rename the domain. A newly created domain contains empty Publish and Subscribe sets.

Renaming a domain

To rename a domain, select the domain in the Explorer View then double-click or press the "F2" key. The Domain name becomes editable, enter the new Domain name and press the "Enter" key. If the domain name is not unique, an error message is displayed and the domain name is reverted to its previous value. Note that all leading and trailing spaces are stripped from the domain name.

Opening an existing DDF

Select the "File->Open" item on the toolbar menu, this causes a dialog window to appear. Select the DDF you want to open and click the "Open" button on the dialog panel. The DDF is parsed (checked for correct syntax), and its content is displayed in the Explorer View. The title bar of the DDF Configurator displays the name of the file that you just opened. If the DDF could not be processed successfully, an error message is displayed and the opening process is aborted.

Creating a data item

Data items are created inside the Publish or the Subscribe set of a domain. There are 2 ways to create a new data item:

- right-click either the Publish or the Subscribe set of a domain, this causes a pop-up menu to appear. Select the "Add Item" item on the pop-up menu,
- select either the Publish or the Subscribe set of a domain, then select the "Item->Add New Item" item on the toolbar menu.

The default data type for a new data item is double. Once an item has been created, you can add a description for it in the Properties View. This description will be saved inside a DDF so that it can be displayed again the next time the DDF is open. A newly created item is automatically assigned the name "new Item#", where "#" represents a positive integer that increases sequentially. Note that data items are sorted in lexicographical (alphabetical) order when they appear inside a data set.

Renaming a data item

Select the data item to rename, then double-click or press the "F2" key. The data item name is editable, enter the new data item name and press the "Enter" key. If the new name doesn't match a name existing in the domain, the data item is renamed, otherwise an error message is displayed and the data item name is changed back to its previous value. Note that all leading and trailing spaces are removed from the item name.

Adding a text description for a data item

To add a text description to a data item, select the item in the Explorer View, this will show the Properties View for the item. Enter a text description under the field entitled "Description For Item [item name]", then press "Enter".

Adding a default value for a data item

To add a default value to a data item, select the item in the Explorer View, this will show the Properties View for the item. Enter a default value under the "Default Value For Item [item name]", then press "Enter". The value is accepted if the format is acceptable as a C code definition of a double precision float.

Note that a default value can only be assigned to a data item that belongs to the Subscribe set of a domain (namely the CLIENT->FRAMEWORK set). In addition, the type "byte" doesn't support default values.

Saving the content of the Explorer View

If the Explorer View contains data read from an existing DDF, the current content of the Explorer View is saved to that same DDF by selecting the "Save->File" item on the toolbar menu. If no DDF has been opened, the current content of the Explorer View can be saved to either a new DDF or a new CSV file. To do so, select the "Save As" menu item from the "File" menu, then choose the desired file format.

Importing an existing DDF

Select the "File->Import" item on the toolbar menu, then choose the "DDF" item. A dialog allows you to pick the DDF. The DDF is opened and parsed, then it is added to the current Orchestra container. If the imported DDF contains a domain whose name matches another domain already present in the Explorer View, the imported domain is rejected and processing proceeds with the remaining domains of the DDF.

Importing a Dymola input file

Select the "File->Import" item on the toolbar menu, then choose the "dsin.txt" item. A dialog allows you to pick the Dymola input file. The file dsin.txt is opened and parsed, then its content is added to a default menu named "dsinImportDomain#", where "#" represents a positive integer that increments sequentially until there is no name conflict with any already defined domain. The inputs to the Dymola

model are inserted into the Publish set (namely the FRAMEWORK->CLIENT set), and the outputs generated by the Dymola model are inserted into the Subscribe set (namely the CLIENT->FRAMEWORK set).

Importing from a CSV file

To import a CSV file, Select the "File->Import" item on the toolbar menu, then choose the "CSV File" items.

Clearing the Explorer View

Select the "File->Clear" item on the toolbar menu to delete all the data contained inside the "Orchestra" root node. Note that this will not remove a previously saved DDF.

Selecting a domain and data items

When you launch the DDF Configurator from a Simulink block of the RT-LAB Orchestra library, not all the domains and data items present in the Explorer View will be mapped back to the block's mask. Only one domain can be associated with a given block, and therefore you need to select that domain if there are more than one domains in the Explorer View. To select a specific domain, left-click it and set the toggle button to "yes". At any time, you can view which domain is currently selected by clicking the "Orchestra" root node, and checking the "Selected Domain" field of the corresponding Properties View. Note that selecting a domain automatically un-selects the previously selected domain, if any.

Once a domain is selected, you can also choose to map only a subset of the data items to a block's imports and outputs. By default, all data items created inside a data set are selected. To un-select a data item, click the item name in the Explorer View, this will display the Properties View for that item. Then choose "no" under the "Item [item name] Is Selected" field on the Properties View.

When DDF Configurator is launched as a standalone application, selecting a domain or selecting data items has no incidence on the generated DDF.

DDF Configurator Command Line Arguments

To launch the DDF Configurator from the command line, enter "ddfConfig.jar" or "java -jar ddfConfig.jar". Either of these commands will show an empty DDF Configurator. The difference between these 2 commands is that the former version does not associate a console with the executable, therefore any error reporting will not appear on the screen. For this reason, it is always preferable to use the latter version of the command when launching DDF Configurator from the command line.

In addition, you can pass options to the DDF Configurator as command line arguments. The following command formats are supported:

- DDFConfig.jar -open sourceFile.[xml,csv,txt] - This command imports a DDF, CSV or Dymola input file and starts a DDF Configurator that shows the corresponding data.
- DDFConfig.jar -convert sourceFile.[xml,csv,txt] targetFile.[xml,csv] - This command imports a DDF, CSV or Dymola input file and converts it to either a DDF or a CSV file. Note that this command will not show the graphical interface.

Again, to execute the DDF Configurator with a console you need to prepend "java -jar" to the above commands.

RT-LAB Orchestra Simulink blocks

RT-LAB Orchestra blocks are connection end points of the Orchestra communication layer, they are used to bridge the Framework and External Components.

There are 2 types of blocks in the RT-LAB Orchestra Simulink library, namely Proxy blocks and External Components blocks. You insert Proxy blocks into a subsystem of an RT-LAB Framework, whereas External blocks are used from within an external Simulink model that you want to connect to an RT-LAB Framework. Refer to the [Blocks reference on page 1](#) chapter of the RT-LAB library's documentation for a detailed description of these blocks.

Setting up an RT-LAB Orchestra co-simulation

Using the Orchestra RT-API

Once you have defined the data items to be exchanged between an external component and the RT-LAB framework in a DDF, you need to embed calls to the Orchestra RT-API into the external component. The Orchestra RT-API is a C code Application Programming Interface (API) that implements a data-centered communication mechanism between co-simulation components. This mechanism allows domain participants to exchange data by referring to it by name only.

Prior to exchanging data with the RT-LAB Framework, a connection to a domain must be established by calling the `RTConnect()` function of the Orchestra RT-API. Data items are published (sent to the RT-LAB Framework) and subscribed to (received from an RT-LAB Framework) by calling `RTPublish()` and `RTSubscribe()` respectively, referring to data items by name. The data exchange with a domain can be synchronous or asynchronous, as specified by the Synchronicity QoS defined in the DDF. For a synchronous connection, calls to `RTPublish()` and `RTSubscribe()` must be enclosed between calls to functions that provide a locking mechanism. Disconnection from a domain is achieved by calling `RTDisconnect()`. Disconnection can happen at any time after an external component has connected, without affecting the RT-LAB Framework.

Refer to chapter [Functions reference on page 3](#) of the RTAPI Reference Guide for a description of the functions that are part of the Orchestra RT-API.

Integration of external C code to an RT-LAB model

This section details the steps required to connect some C code to an RT-LAB Framework

- Add a C Proxy block to a subsystem of the Framework. Configure the block by double-clicking on it, which opens the DDF Configurator graphical tool. Specify a domain name and define a list of signals to be exchanged with the legacy C code. Select items in the Publish set ("Framework-->Client") to send the corresponding signals to external domain participants. Reciprocally, select items in the Subscribe Set to receive data from external domain participants.
- Build and load the RT-LAB Framework like any other RT-LAB model.
- In the legacy C code, add calls to connect to the domain and publish/subscribe calls to send signals to the RT-LAB Framework and receive signals from it respectively.
- Compile and link the C code into a standalone process.

A C code sample is provided in the RT-LAB distribution under the directory <RT-LAB installation dir>\Examples\Features\Orchestra\C_Code\src.

Integration of an external Simulink model to an RT-LAB Framework

- Add a Controller block anywhere inside the external Simulink model.
- Add Publish and Subscribe blocks where you want signals to be sent to and received from the RT-LAB Framework.
- Build the model by using Simulink RTW.

A sample Simulink model that connects to an Orchestra co-simulation is provided in the RT-LAB distribution under the directory <RT-LAB installation dir>\Examples\Features\Orchestra\Simulink\src.

Integration of a Dymola model to an RT-LAB Framework

This section details the steps required to connect a standalone Dymola application to an RT-LAB Framework.

- Translate the Dymola model you want to connect to an RT-LAB Framework. This will generate a dsmodel.c file, and a dsin.txt that contains the lists of inputs and outputs signals to the Dymola model.
- In a subsystem of the RT-LAB Framework, add a Dymola Proxy block. Configure the block by double-clicking on it, which will open the DDF Configurator graphical tool. Import the Dymola dsin.txt file and select the input and output signals.
- Add calls to the Orchestra RT-API in file StandAloneDymosim.c. Build an executable for the standalone Dymola model. Refer to <RT-LAB installation dir>\Examples\features\Orchestra\Dymola\src for a list of source files required to build this executable.

Running an RT-LAB Orchestra co-simulation

This section describes how to set up the External Components you want to connect to an RT-LAB Framework. It is assumed that you have already built and loaded the Framework itself.

Transferring External Components executables

Build an executable for each external component you want to connect to an RT-LAB Framework by following the steps listed above. Note that when setting up an RT-LAB Orchestra co-simulation, domain participants need to be co-located on the same processor. Specifically, the executable for an external component that connects to a given domain must reside on the same target node as the RT-LAB subsystem that contains the Proxy block for that domain. Open an FTP connection to a target node and copy the External Component's executable. Run the executable by typing its name inside of a command window.

Environment requirements

RT-LAB Orchestra is supported on the same platforms as RT-LAB.

To run the DDF Configurator as a standalone application, you need a Java J2SE Runtime Environment 5.0 or later (JRE) installed on your machine. The JRE can be downloaded free of charge from Sun's website (<http://java.sun.com>). To run the DDF Configurator from a Simulink block, you need to use Matlab 7.1 (Matlab R14 SP3) or later.

RT-LAB Asynchronous processes

Introduction

In keeping with the design principle of openness, RT-LAB features can be extended through the use of Asynchronous User Applications also called Asynchronous Processes. This gives RT-LAB users access to the full power of the operating system and allow them to implement their own interfaces to various external devices. Although usually targeted for use with communication devices such as GPS receivers and power monitors, Asynchronous User Applications can be developed to interface acquisition boards, or implement system specific software such as file management, etc.

Architecture overview

Asynchronous User Applications allow the user to execute applications asynchronously to RT-LAB, while running on the same target computer as RT-LAB. They are asynchronous because they are not synchronized with the model, i.e. they run independently of the RT-LAB model, either at their own periodic rate or triggered by an external interrupt.

To illustrate this, let's take example on an application that needs to perform data acquisition on a serial link at low speed, while processing other computation data at higher speed. It would not be convenient in that case to run the model at the low speed of the serial link. In the Asynchronous Process architecture, the low speed serial data acquisition is performed in a process separate from the main RT-LAB model, the Asynchronous Program. The RT-LAB model can then be set up with the calculation step required to process the high speed data. Dedicated Simulink blocks are then placed in the RT-LAB model in order to allow data transfers to and from the Asynchronous Program.

One Asynchronous Process application thus consists in two parts:

1. One source code written in C or C++ implements the user specific function and handles communication with the RT-LAB model. This source code is compiled in an executable program which is transferred to the target computer and launched during model initialization,
2. A set of blocks placed inside the RT-LAB model allows the user to specify the configuration parameters of the Asynchronous Program, and to select the data signals to be sent to, or retrieved from, the Asynchronous Program during execution of the model.

Note that although the source code of the Asynchronous Program must be designed specifically for the user application using templates and the API of functions provided with RT-LAB, Simulink blocks usually do not require development other than possibly masking the basic blocks provided with RT-LAB to make a specific user interface for the application.

Asynchronous Programs exchange data with the model periodically via a shared memory opened by the RT-LAB model during initialization as presented in [Figure 15](#) below. RT-LAB then starts the asynchronous program which name is defined in a configuration block placed in the RT-LAB model, the Asynchronous Controller block. During model execution, both the model and the Asynchronous Process read from, and write to, the shared memory in order to exchange data.

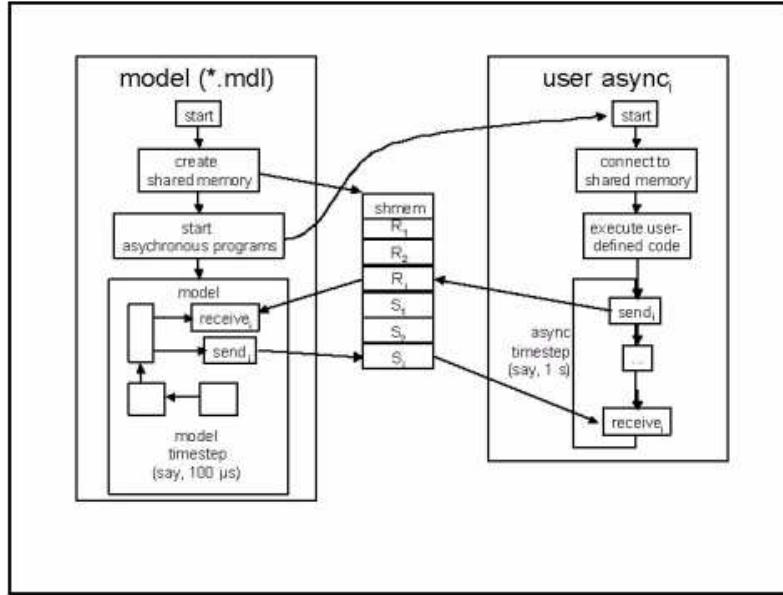


Figure 14:Communication between asynchronous user program and RT-LAB model.

In the serial communication example described above, data are processed in the Asynchronous Program as soon as they are received on the serial link, and are placed in the shared memory at specific locations determined by the developer of the Asynchronous Program. Asynchronous receive blocks placed in the RT-LAB model periodically access the shared memory and return the data present in the shared memory via the outports of these blocks. The data can then be used within the RT-LAB model for further processing. In a similar way data can be transferred from the RT-LAB model to the shared memory by the use of Asynchronous Send blocks. These data can then be retrieved by the Asynchronous process in the shared memory and transmitted to the serial link.

Building the asynchronous application

Preparing the RT-LAB model

The blockset of dedicated Simulink blocks can be found in the Simulink library browser in the section: *RT-LAB->Communication->Asynchronous* as presented in Figure 16 below

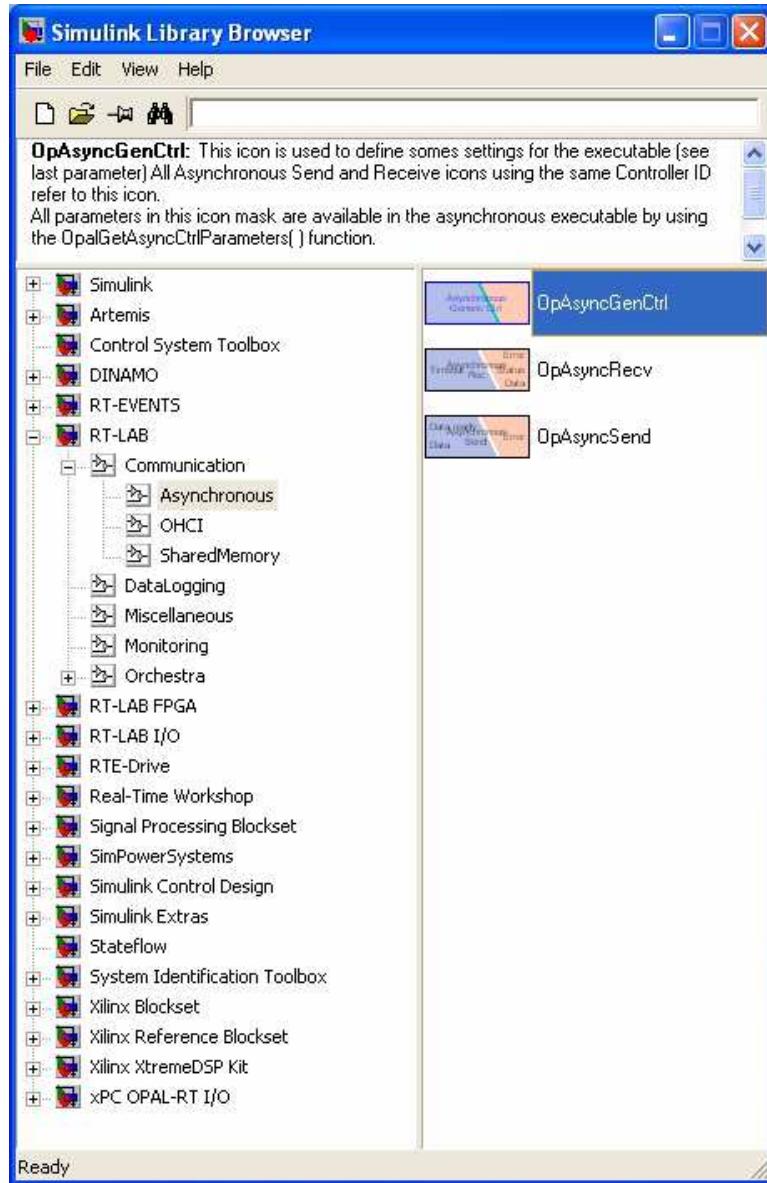


Figure 15:RT-LAB Asynchronous Program library in Simulink browser

This library consists in three blocks: **OpAsyncGenCtrl**, **OpAsyncRecv** and **OpAsyncSend**. We will describe their main functionalities here. Their full description can be found in the Block library section of the RT-LAB documentation.

- OpAsyncGenCtrl

In order to interface the RT-LAB model with an Asynchronous Program running on the target computer, at least one OpAsyncGenCtrl block must be placed in the model. The OpAsyncGenCtrl block is used to define the configuration settings of the Asynchronous Program created by the user.

This block controls the launch of one specific instance of the Asynchronous Program to be run on the target where the model executes. The name of the Asynchronous Program is specified via a mask parameter, and the Asynchronous Program itself must be compiled and transferred to the target prior to model execution. The following sections will detail the build and transfer process.

Several OpAsyncGenCtrl blocks can be placed in the same RT-LAB model to control either several instances of the same Asynchronous Program, or several different Asynchronous Programs.

Each OpAsyncGenCtrl block defines one unique Controller ID that is used by the asynchronous Send and Receive blocks to refer to the Asynchronous Program instance controlled by this OpAsyncGenCtrl block.

The OpAsyncGenCtrl block also allows the user to specify a set of parameters to be transferred to the program. These parameters are retrieved by the Asynchronous Program using the OpGetAsyncCtrlParameters() function of the asynchronous programs API.

- OpAsyncRecv

This block is used to receive data from the Asynchronous Program during model execution. Its input allows the user to specify a Timeout value to be optionally used by the Asynchronous Program. The Data output contains the values retrieved by the model in the shared memory. It is possible to use a demux block to receive multiple values.

- OpAsyncSend

This block is used to transfer data from the model to the Asynchronous Program. The input is the data to be copied to the shared memory. It is possible to use a mux block to send multiple values. Data is actually updated in the shared memory when the Data Ready line is asserted.

The figure below is an example of usage of these three blocks in the example model for TCP/IP communication.

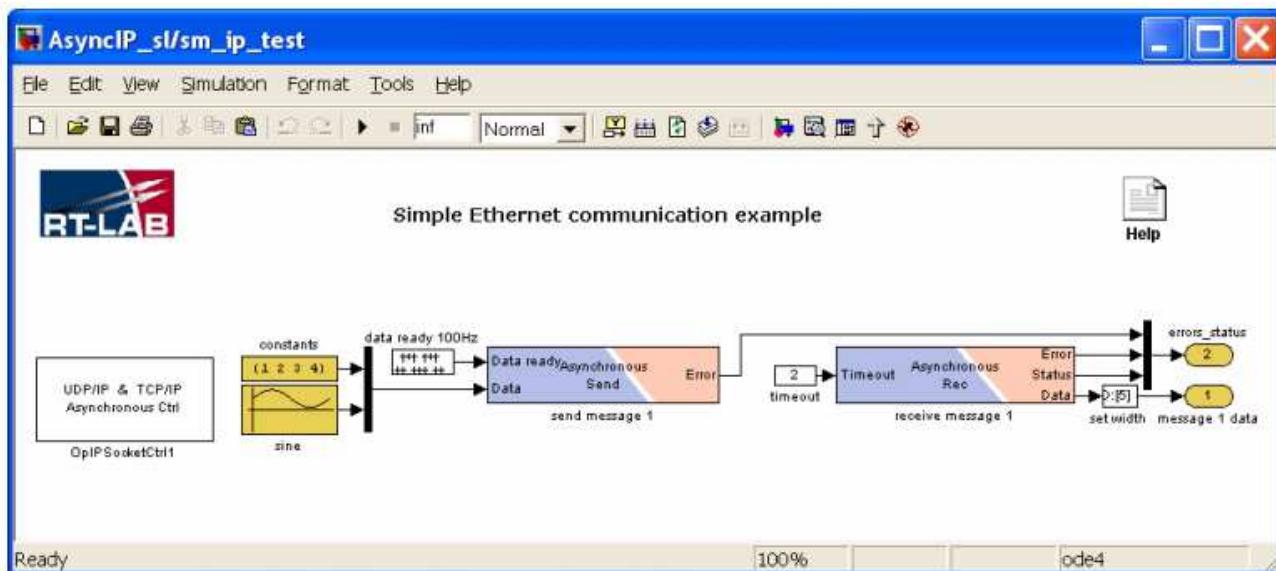


Figure 16: Example model subsystem with RT-LAB asynchronous blocks

Note that in this example, the OpAsyncGenCtrl is replaced by the OpIPSocketCtrl block which uses the same s-function sfun_gen_async_ctrl as the OpAsyncGenCtrl block but with a mask adapted to present only the subset of configuration parameters used by this specific application.

Preparing the C source code

Once the user has defined what signals of his model must interface with the Asynchronous Program, he must develop the C source code that will implement the specific algorithm required by his application.

All Asynchronous Programs use minimally the same 4 steps: Initialization, send loop, receives loop and close section. All of these programs use the Asynchronous API functions provided with RT-LAB. The available API functions are listed in the header file AsyncApi.h located in folder **\$RT-LAB\$\common\include_target**.

Note that RT-LAB provides two complete examples containing the models and their associated Asynchronous Program C code. These examples are available as templates for RT-LAB Projects:

- **IO/_Generic/_async_ip** is an example of UDP/IP and TCP/IP communication,

-
- **IO/_Generic/_async_serial** is an example of serial communication.

See [New RT-LAB project wizard](#) to learn about RT-LAB templates.

The C code of these examples is commented step-by-step and indicates the sections where the user can insert his application-specific code. They can be used as templates for other applications.

Preparing the makefile

In order to compile this source code into an executable program, the user must also prepare its associated makefile. It is assumed here that the reader is familiar with setting up C makefiles. The above-mentioned templates also provide the makefile associated with the C source code. See for example /async_serial_source/AsyncSerial.mk in the Async_serial example.

The libraries libOpalAsyncApiCore.a, libOpalCore.a and libOpalUtils.a contain all the definitions required to link C source code using the Asynchronous Process API functions, so the LIBS definition should look like:

```
LIBS = -IOpalAsyncApiCore -IOpalCore -IOpalUtils -IOpalCore
```

Note the repetition of -IOpalCore in the list, due to file dependencies in the libraries.

- In Simulink environment, libraries must be transferred to the model directory on the target prior to model compilation

RT-LAB takes care of transferring core libraries such as libOpalCore.a and libOpalUtils.a to the model directory during the build process. The library libOpalAsyncApiCore.a however is not systematically transferred by RT-LAB since it is required only for linking Asynchronous Programs.

RT-LAB models using Asynchronous Process controller blocks provided in the RT-LAB I/O library already take care of transferring all required libraries by the use of Matlab m-file callbacks. Users who have disabled the link of their controller blocks to their original RT-LAB library blocks in their own models need to restore this link in order to retrieve the proper callbacks settings.

Users who have made their own Simulink mask above the s-function sfun_gen_async_ctrl need to add the libOpalAsyncApiCore.a library to the list of files to transfer to the target. This will be explained in the next section.

Furthermore, in order for the makefile to link with the libraries found in the model directory, the model folder must be added to the library path by adding the option '-L.' to the link flags.

Configuring RT-LAB for build of the asynchronous program

After preparing both the Asynchronous Program source code and RT-LAB model, to ensure that the application is properly compiled, follow these steps:

1. In RT-LAB, open a **Model Editor** and go to the [Files Page](#).

This page allows the user to specify what files will be uploaded to the target and what files will be retrieved from the target:

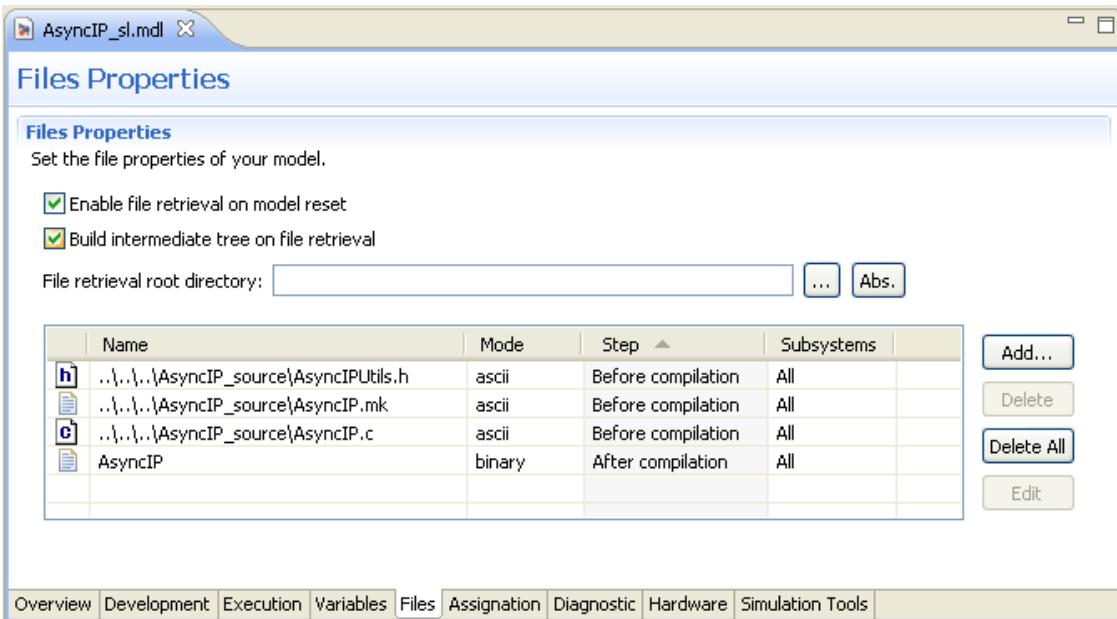


Figure 17: Specifying file transfer at compilation time

File names may be full-qualified, i.e. path + filename, or may be relative to the model folder.

As explained in the previous section, the library libOpalAsyncApiCore.a may need to be added to this list if it is not added to the build process in a Matlab callback of the controller block. The path to this library is (replace \$RTLAB-ROOT\$ by the actual path to the RT-LAB root folder):

\$RTLAB_ROOT\$/common/lib/qnx6/libOpalAsyncApiCore.a when compiling for QNX6 targets,
or
\$RTLAB_ROOT\$/common/lib/redhawk/libOpalAsyncApiCore.a for Red Hat targets.

This library must be transferred in binary mode.

During model initialization, the OpAsyncGenCtrl block will attempt to launch the Asynchronous Program. For this reason the executable file of the Asynchronous Program must be found in the model directory.

If the execution node is different from the compilation node, the executable file of the Asynchronous Program, which was created in the model directory during compilation, needs to be transferred back to the host computer. This is done by adding this file to the list of files to be retrieved from the target after compilation.

This executable file must be transferred in binary mode.

2. On the Development page, under the Compiler option tab, specify the following command in the Compiler command text box as shown in Figure 19. The compilation command is:

make -f /usr/opalrt/common/bin/opalmodelmk (for QNX targets)
or
/usr/bin/make -f /usr/opalrt/common/bin/opalmodelmk (for Red Hat targets)

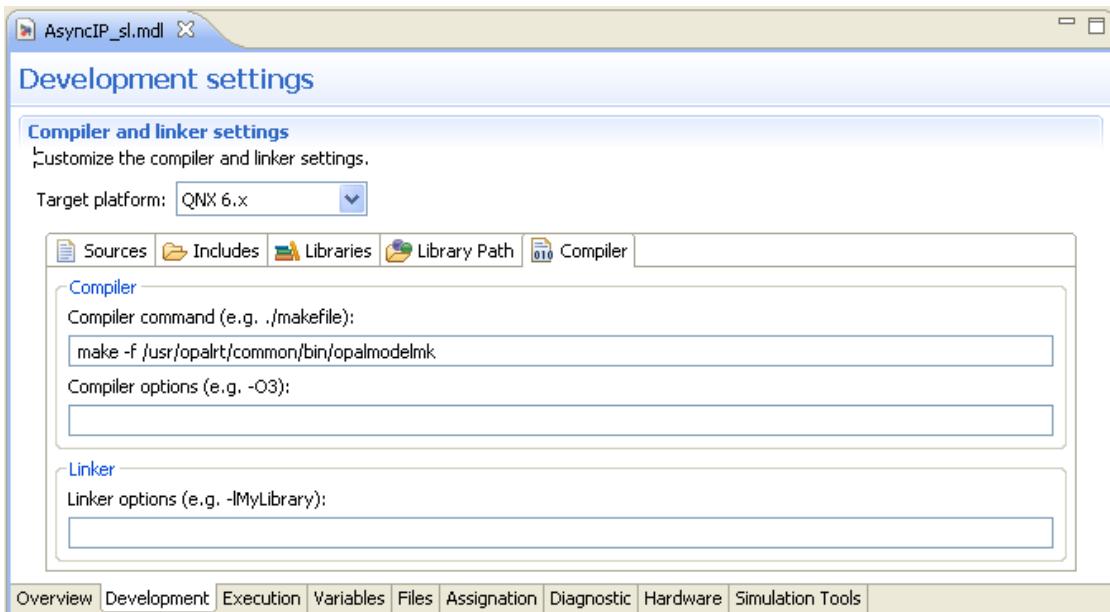


Figure 18:Setting the compilation command

You are then ready to build the model. Note that if there is a problem in the C file of the Asynchronous process, errors will be displayed in the **Compilation View** during compilation.

Running the Asynchronous program

Loading the model

The executable file of the Asynchronous Program must be added to the list of files to be transferred to the execution node at load time as shown in the next figure:

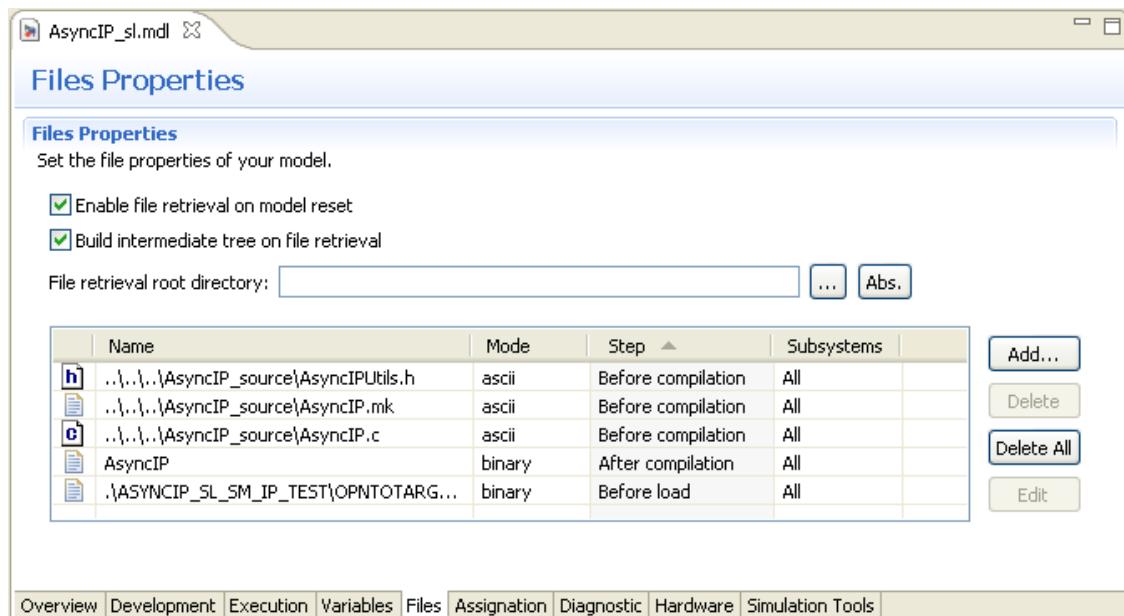


Figure 19:Specifying executable transfer at load time

After retrieval from the compilation target, the executable file is found in the subsystem sub-directory created in the model directory of the host computer during compilation. Remember that the executable must be transferred in binary mode.

The model is then ready to be loaded by clicking the Load button of the RT-LAB **Main Toolbar**. As mentioned above, the sfun_gen_async_ctrl s-function (used by the OpGenAsyncCtrl block or the replacement controller block developed for the specific application) will attempt to launch the Asynchronous Program during initialization of the model. Note that because the Asynchronous Program is asynchronous to the model, its own initialization phase may span longer than the model initialization.

Messages prepared by the Asynchronous Program are displayed in the **Display View** together with messages from the model.

Executing the model

During model execution data are exchanged with the Asynchronous program via the shared memory. The Data Ready of the OpGenAsyncSend block specify when to initiate a data transfer towards the Asynchronous Program. The OpGenAsyncRecv blocks return data placed by the Asynchronous program in the shared memory. The developer of the Asynchronous Program specifies how these data are handled by the Asynchronous Program.

Resetting the model

During model reset, RT-LAB attempts to close the shared memory opened during model initialization. In order to be able to do so, the model will first wait for the Asynchronous Program to exit. The Asynchronous Program is notified that the user has requested a Reset by checking the MODEL_STATE variable. Upon detecting a Reset, or an Error state, the Asynchronous Program should exit properly. If it fails to do so, for example because it is too busy processing data from an external device at that time, RT-LAB model may fail to close the shared memory and will display a message to warn the user about this. The user may then need to kill the Asynchronous Program manually on the target computer before the next load of the model. It is the responsibility of the Asynchronous Program developer to ensure that Reset or Error states are handled properly by the Asynchronous Program.

RT-LAB User SFunction

Introduction

Prior to Release 13 of Real-Time Workshop, the RT-LAB targets used the SimStruct data-structure to capture and store model-wide information. Since the SimStruct was also used by noninlined S-functions, it suffered from the drawback that some of its fields remained unused when it was used to capture root (model-wide) information. To avoid this drawback, Version 5.0 of RTW introduces a special data structure called the rtModel to capture root model data.

RT-LAB now supports the data structure rtModel. Support for this structure will help make RT-LAB compatible with new MatLab toolboxes, such as Simscape toolbox with SimDriveline, SimHydraulics, SimMechanics and will give access in the future release of RT-Lab to new RTW functionality, such as support of Model Reference Block ...etc

For most users, this new structure should have no impact on the use of RT-Lab. Only users who have created their own libraries from S-Function will need to build their libraries with a new compilation flag.

Changes Resulting from the Replacement of SimStruct with the rtModel

The implications resulting from the use of this new structure will apply only to users of Simulink S-Function.

Using Simulink Sfunction consists of three parts:

- User of Sfunction built with RT-LAB model: No modification required.
- User of external Sfunction included in external library: these users need to compile their library with a new compilation flag.

In order to compile the Sfunction source code for Rtmodel support, the user must also prepare its associated makefile. It is assumed here that the reader is familiar with setting up C makefiles. A new compilation flag "**USE_RTMODEL**" should be added to support rtmodel data structure.

RT-LAB / Xilinx System Generator toolbox integration

Starting with RT-LAB v8.3, the integration between the Xilinx System generator toolbox and RT-LAB requires the use of the new Opal-RT RT-XSG product. RT-XSG provides all files required for preparing Simulink models that can be built into FPGA bitstreams for the Opal-RT products. RT-LAB only contains the files and drivers needed for interacting with these Opal-RT boards during the real-time simulation.

RT-LAB examples models for real-time simulations integrating Opal-RT boards supported by RT-XSG can be found under the <RT-LAB_ROOT>\Examples\IO\Opal-RT folder. These examples are thus 'CPU models', i.e models that run on the CPU of the target machine to which the board is connected. The structure of these models is described in the following sections. The OP5130-XSG subdirectory, for example, contains one basic example model that performs data exchange with an OP5130 board programmed with a bitstream generated with RT-XSG. The generated bitstream is provided with this example model and is loaded into the FPGA of the OP5130 at model load time. However, if the user wants to modify the logic implemented in this bitstream, he must use the RT-XSG product, edit the basic Simulink-XSG example model for this board provided in the RT-XSG folder, regenerate the bitstream, and then copy it to the RT-LAB example model folder.

Requirements

The section in [Chapter 3](#) gives the compatibility matrix of RT-LAB and RT-XSG products.

Preparing CPU models

It is assumed that the reader is familiar with the Xilinx System Generator as presented in Xilinx SysGen user guide. In the following, the discussion focuses on the use of the Opal-RT OP5130 board, although the process of preparing models for other supported boards will be similar.

When designing and XSG-based RT-LAB application, two Simulink models must be developed. The first one, hereafter called the FPGA model, contains the XSG and RT-XSG blocks required to build the bitstream to be downloaded in the FPGA-chip of the reconfigurable I/O card.

The second model, the CPU model, runs on the target node and must contain one interface block, the OpCtrlReconfigurableIO block, in order to manage communication between the CPU model and the reconfigurable I/O card. This block can be seen as a bridge between software and hardware. It communicates and receives in real-time the data samples to and from the OP5130 reconfigurable IO card through the Opal-RT SignalWire communication link. An example CPU model is presented in [Figure 21](#). In this example, the OpCtrl ReconfigurableIO block is set up with 16 IN/OUT communication ports, thus allowing transfer of 16 32-bit values of data to and from the OP5130 card at each calculation step. The number of inports and outports of the OpCtrlReconfigurableIO block is configurable between 0 and 16. Note that the width of each inport and outport can also be increased up to 250 32-bit values if required by the application.

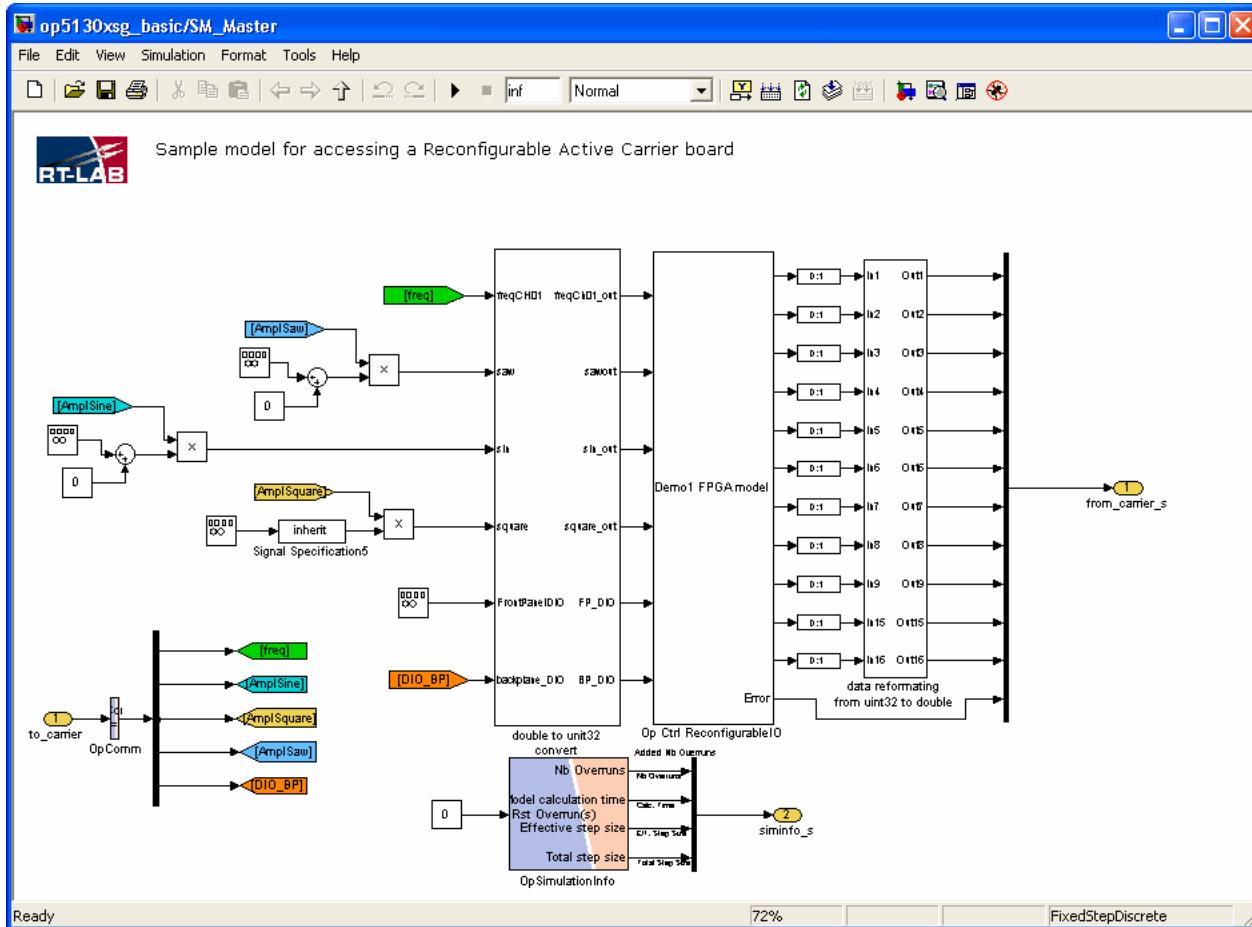


Figure 20: Example of a CPU model

In this particular example, the imports of the OpCtrlReconfigurableIO block placed in the CPU model are connected to signal generators that generate samples to be transmitted to the OP5130 at a rate of $T_s=200\mu s$. These signal generators create a saw, a sin and a square waveform. Notice that these signals pass through a subsystem named "double to uint32 convert" that converts the generators double type signals to the uint32 type. This is the default type supported by the OpCtrl ReconfigurableIO on either its inputs or outputs. Besides doing signal conversion, the subsystem does signal scaling and concatenation. Scaling is necessary before the type conversion so that decimal values are not truncated. In this particular case, the waveform signals are routed to a DAC interface in the FPGA XSG model which expects the Xilinx Fix16_11 format. So a "Shift Arithmetic" block shifts the three waveform signals by 11 bits to the left (i.e multiply them by 2^{11}).

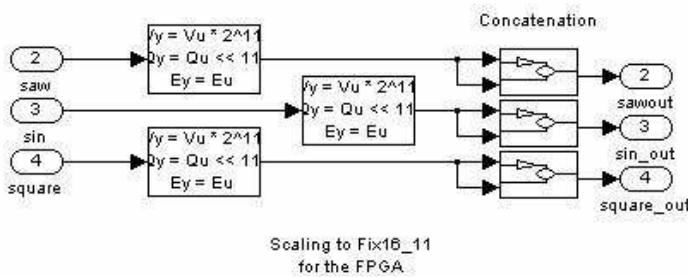


Figure 21:Conversion subsystem connected to the inputs of the OpCtrlReconfigurableIO block.

Inside the three Concatenation subsystems you will find the type conversion blocks as well as the concatenation logic. Concatenation is necessary in this case because the waveform generators are connected to a DAC I/F with 16 bit channels.

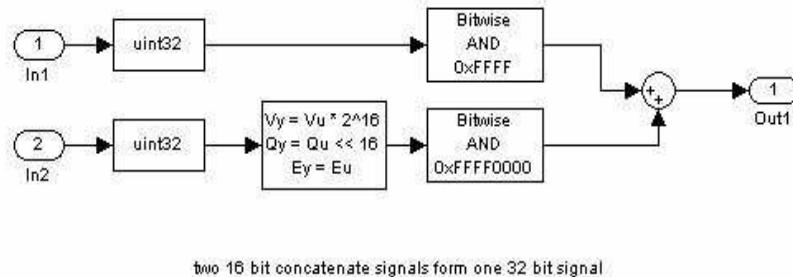


Figure 22:Concatenation of 2 16-bit data values into one single value to be transmitted to the OP5130

Input port In1 takes care of the lower 16 bits and input port In2 of the upper part. In this example, both ports are connected to the same source and come out of output port 2, sawout, of the "double to uint32 convert" which in turn is connected to the input port number 2 of the OpCtrl ReconfigurableIO block. This port corresponds to one entry port of the FPGA model, which, in this example, is directly connected to the OP5330 DAC interface block. Each of the DAC input port of this block represents two 16 bit concatenated channels.

Signal concatenation as presented above is not mandatory and will depend on the logic implemented in each FPGA model, but it is nonetheless advantageous because it uses the available bandwidth more efficiently.

The outputs of the OpCtrlReconfigurableIO block are used to retrieve data from the OP5130 board at each calculation step. The samples are transmitted to the target node via the SignalWire link. Here again some transformation of the data may be needed. The figure below shows the type of transformation done in the "data reformatting from uint32 to double" subsystem to extract 16-bit ADC samples and convert them to the double type. Note that a shift arithmetic block is used to scale the information properly. Since ADC samples are in the Fix_16_10 format in the FPGA, a shift by 10 bit to the right is necessary in the target.

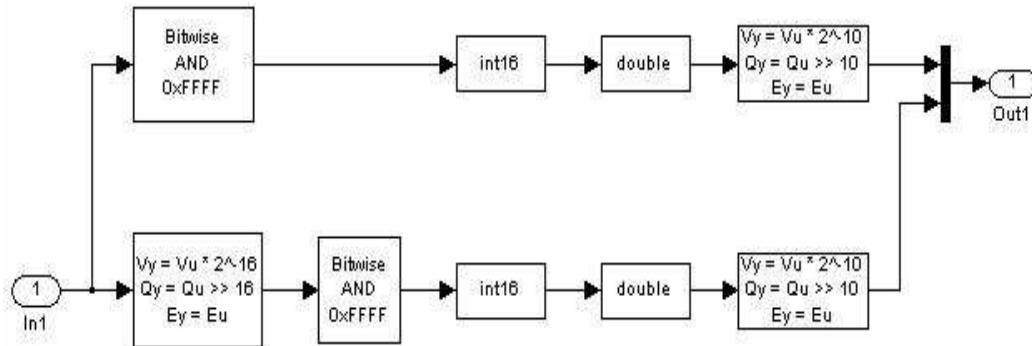


Figure 23:Conversion subsystem connected to the inputs of the OpCtrlReconfigurableIO block.

Limitations

- As of RT-LAB v8.3, the only reconfigurable IO card supported is the Opal-RT OP5130 card. RT-LAB support for new RT-XSG-supported boards such as Xilinx ML506 development board and Opal-RT Spartan3-based module is under way and RT-LAB examples for these boards will be integrated in future releases.
- Communication between multiple OP5130 reconfigurable IO card is not yet possible. However it is possible to place several OpCtrlReconfigurableIO blocks in the model in order to communicate with several OP5130 cards from the same CPU model. The 'Search Strategy' option of the OpCtrlReconfigurableIO block must then be used to differentiate between the OP5130 cards.
- Only 16 32 bit IN/OUT ports are provided by the DataIN/OUT blocks, and the maximum width of each of these outputs is 250 samples.

Using ScopeView

This chapter describes how to use ScopeView with RT-LAB.

Introduction

The ScopeView software is used to display signals and waveforms acquired from various data acquisition systems and to perform mathematical calculations and signal processing. ScopeView is tightly integrated with RT-LAB and allows visualization of signals acquired from real-time simulation and also serves as an analog and digital oscilloscope.

ScopeView can acquire and import signals and data from many sources. It supports RT-LAB data source and other data source types like HYPERSIM, EMTP-RV, MATLAB and, COMTRADE. It also allows managing many data sources at the same time.

Once the data has been acquired and/or imported, ScopeView make it possible to execute many type of graphics processing, such as zoom, superimposition, tracking cursor, graph displacement. It also offers a rich set of mathematical functions to calculate new waveforms and to generate complex analyse. Report can then be produced and exported to various file formats such as PDF, MATLAB or Post-Script.

More information is available in ScopeView user manual; to access this document, open ScopeView and click the **User Manual** item from the **Help** menu.

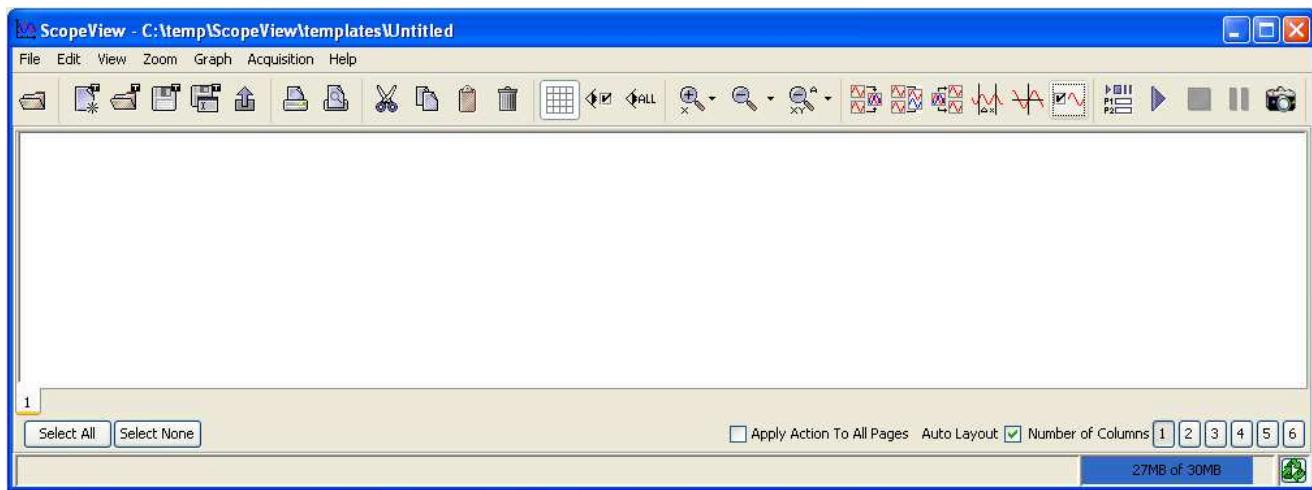
Opening ScopeView

To open ScopeView, click on the ScopeView icon in the RT-LAB main toolbar. This icon is represented by a Cartesian plan with red and blue signals.



ScopeView will appear after the display of the splash screen. Please refer to the RT-LAB installation guide for more details on how to install ScopeView. Note that a valid ScopeView license is required to launch ScopeView.

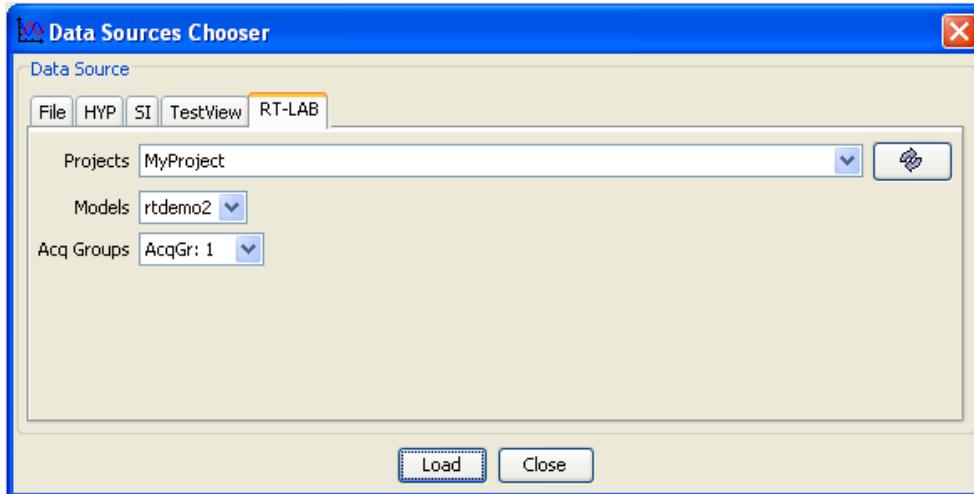
At start-up, ScopeView opens the main ScopeView window and the Signals form. The Signals form allows choosing a signal source. Refers to the following sections for more details on the main ScopeView window and the Signal form.



Selecting and Loading an RT-LAB data source

In order to be able to acquire data and signals from the real-time simulator, you need to select and load an RT-LAB data source in ScopeView. Click the **Open Data Source Files** button from the main toolbar. Alternatively, click the corresponding **File** menu item (**Ctrl+O**).

At this point, the **Data Sources Chooser** dialog is displayed.



Select the **RT-LAB** tab from the dialog and then choose the project and model from which the data should be acquired. Note that this model must be compiled correctly to be available in the list of models and running on the target to perform acquisition operation.

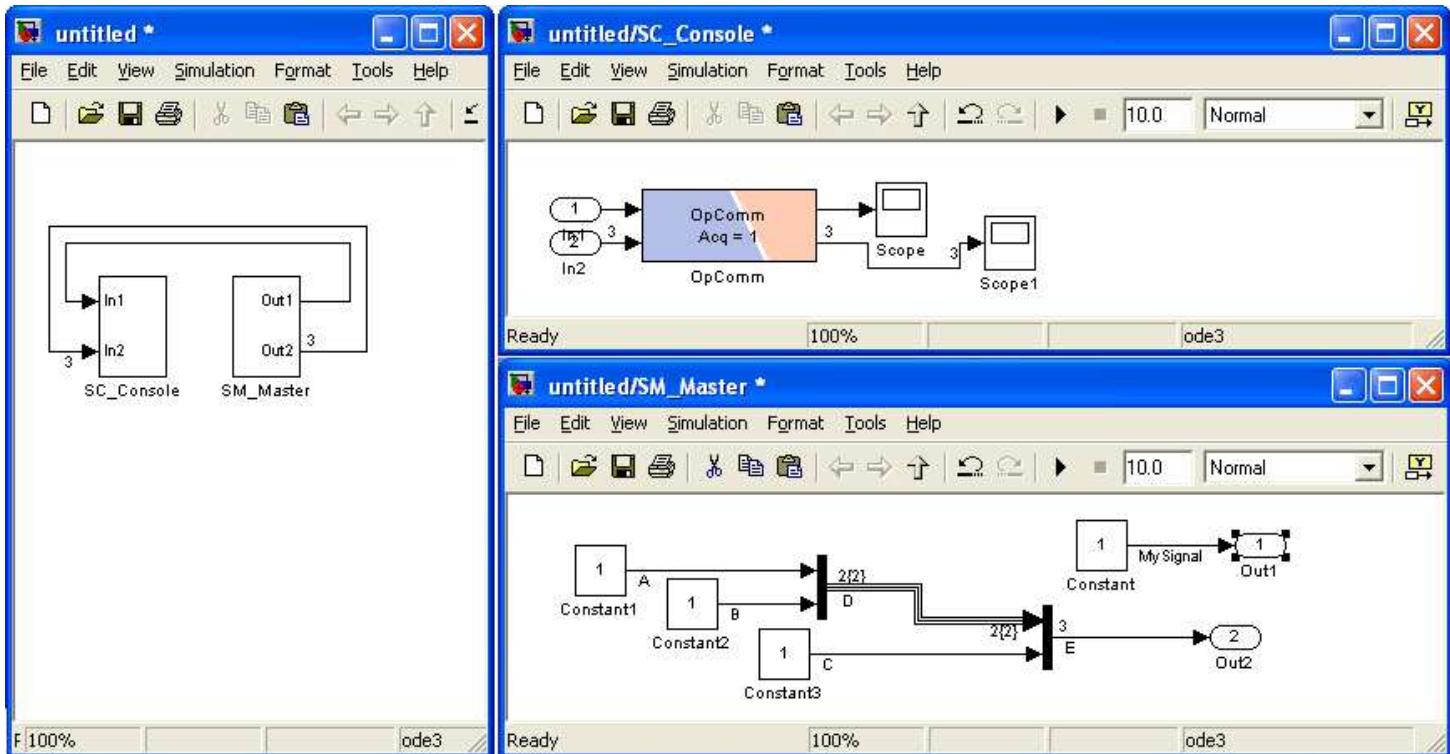
Also select one of the model's acquisition groups and click the **Load** button. All signals from this acquisition group will be available in the Signal Form of ScopeView.

Adding Signals to the Acquisition Group

There are two methods to add signals to an acquisition group. The usual method is based on the model's console:

- First, add one or more OpComm block to your model's console. Each console OpComm refers to one acquisition group.
- Open the OpComm Mask Parameters dialog.
- Change the **Acquisition group number**. This number represents the acquisition group ID. All signals of this group will share the same acquisition parameters.
- Change the **Number of imports** of the block. Each import will receive one or more signals from the real-time subsystem and each signal that passes through this block will be available in ScopeView.
- Close the OpComm Mask Parameters dialog.
- Wire signals from your real-time subsystem to your console's OpComm blocks.
- To change the signal name that will be used by the acquisition, right click the signal's source in the real-time subsystem and select **Signal Properties...** and type a new Signal name. This name will be shown in ScopeView Signal Form. See **Selecting Signals** section for more information.
- If signals that pass through the OpComm are buses, change the signal names after each virtual block (mux, bus creator, ...) of real-time subsystem to create complex name structure.
- Rebuild your model with RT-LAB.

The following figure shows a model that has one acquisition group that contains 4 signals. One signal, called MySignal, passes through import 1 of the OpComm block. Three signals pass through the second import of the OpComm block. The signals' bus name are respectively A.D.E, B.D.E and C.E.



The second method, called dynamic acquisition, allows the user to access any model's signals without having to wire them from the real-time subsystem blocks to the console OpComm block. Here are the steps to add signals using dynamic acquisition:

- Add one or more OpComm block to your model's console. Each console OpComm refers to one acquisition group. Usually, the same OpComm block is used for normal acquisition and dynamic acquisition.
- Open the OpComm Mask Parameters dialog. Set the acquisition group and the number of imports as explained above.
- Check the **Dynamic signals output**.
- Close the OpComm Mask Parameters dialog.
- Make sure your model is properly compiled with RT-LAB
- Open the **Probe Control Panel** of RT-LAB using the RT-LAB main Toolbar.
- Click the tab at the top of the **Probe Control Panel** to select one of the acquisition groups.
- Click the **Signals** button to open the Dynamic Signal Panel dialog.
- Using the **Current model systems and signals set** tree, browse signals, select them and click the Add button. The signal will be added to the **Dynamic signal lists**.
- The dynamic signal names are defined in the real-time subsystem directly on the wire at the output of the block that calculates and outputs the value of the signal. To change the name, as mentioned above, right click the signal in the real-time subsystem and select **Signal Properties...** and type a new Signal name. Rebuild your model.
- Close the Dynamic Signal Panel and close the **Probe Control Panel**.

Refer to the [Acquiring and Viewing Data](#) section of the **User Guide** for more details on how to add signals to the acquisition groups.

Selecting Signals

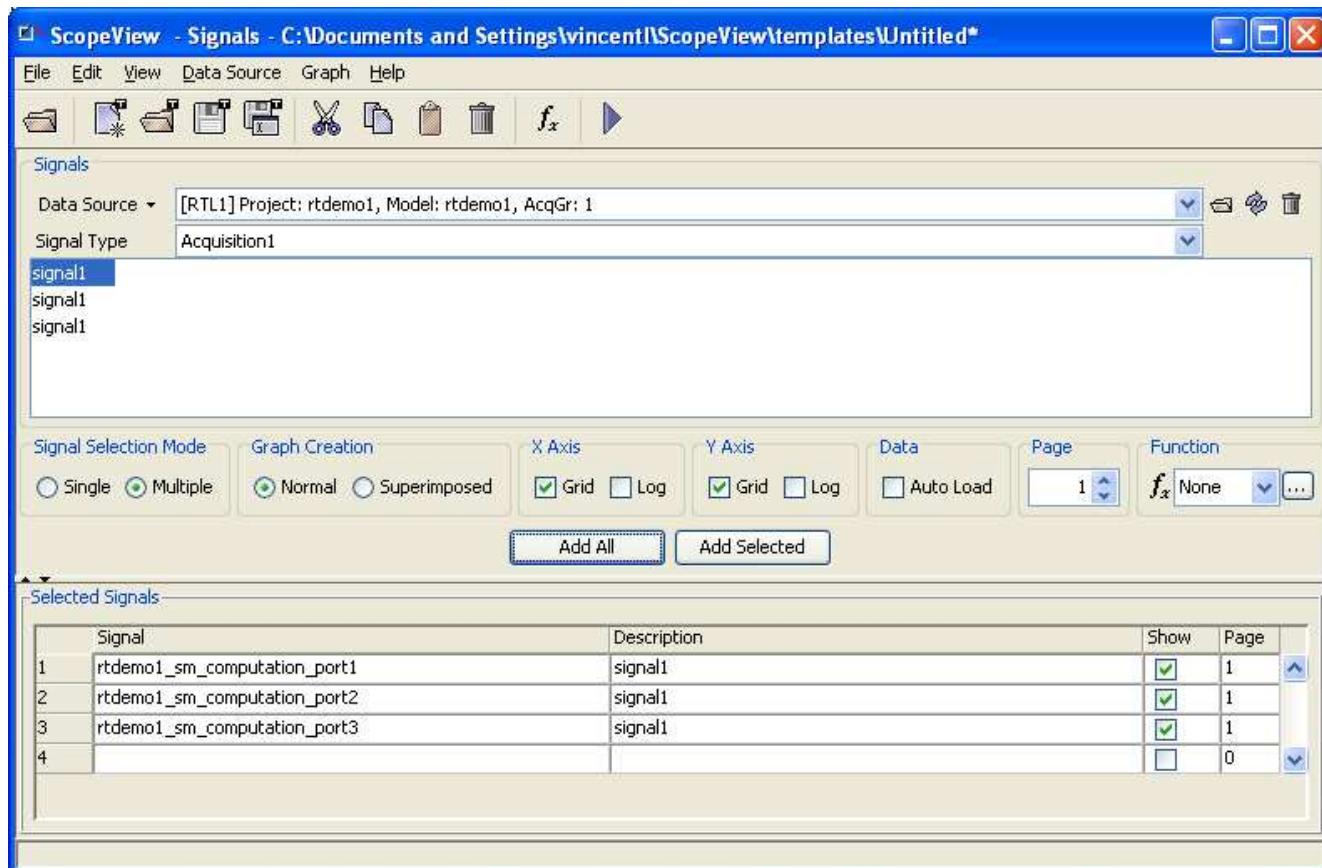
Once the data source is selected and loaded, the **Signals** form lets you choose signals to be acquired for each data source and select the signals that will appear on each graph. It also lets you perform mathematical operations and signal processing.

To open the Signals form, click the **Select Signals** button from the main toolbar. Alternatively, click the **Select Signals** item from the **Graph** menu.



To add signals to a graph, first select the data source from the **Data Source List** and then select the type of signal to be shown in the list. By default, signals sent to model's console are listed, but it is also possible to display dynamic acquisition signals. Also note that the list of dynamic signals available in ScopeView could be selected using RT-LAB [Probe Control Panel](#).

Click the signal's name to add it to the **Selected Signals** at the bottom of the form. These signals are also automatically inserted into graphs of the current report page.



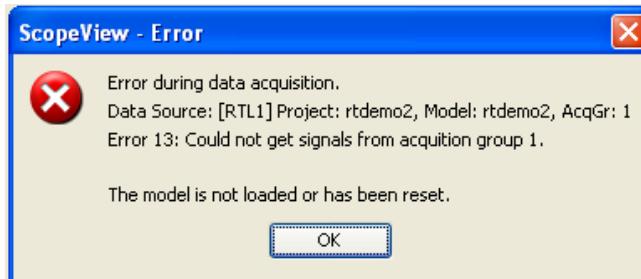
When a large number of signals must be selected, change the **Signal Selection Mode** option to **Multiple** to add multiple signals at the same time. Also change the **Graph Creation** option to **Superimposed** to add multiple signals to the same graph instead of individual graphs.

Advanced options and other functionalities are explained in the ScopeView [User Manual](#).

Acquiring Signals

To acquire signals, first verify that all models used by the RT-LAB data sources are running. Make sure that signals are selected and already assigned correctly to graphs as described in the previous section. Start the acquisition by pressing the **Start** button from the main toolbar. Alternatively, click the **Start** item from the **Acquisition** menu.

At this point, ScopeView starts the process of getting signals from the real-time simulator and displays signals in the graphs. When the model is paused, ScopeView will wait until the model sends its signals. Also note that RT-LAB will display the following error message when the model is not loaded.



Using ScopeView Acquisition Parameters

The ScopeView acquisition parameters are used to configure how the acquisitions are performed. For example, they allow changing the sampling duration, the number of acquisitions to perform or activating synchronization and triggering.



Acquisition parameters are shown under the main toolbar by clicking on the **Acquisition Parameters** toggle button of the main toolbar. Alternatively, these parameters could be displayed by clicking on the **Acquisition Parameters** item of **View** menu.



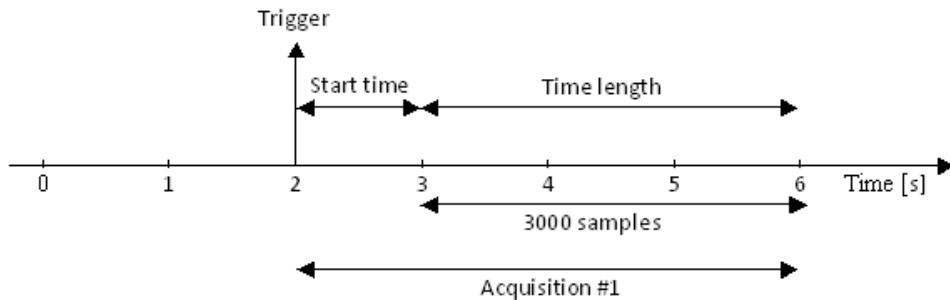
Table 3: Acquisition parameters

PARAMETERS	DESCRIPTION
Start time	Not yet available with RT-LAB data source. Time elapsed before the acquisition started. Use the Probe Control Panel's offset and re-arm delay parameters instead.
Time length	Sampling duration (sampling time length) of one acquisition. Changing this parameter also change the duration parameter of the Probe Control Panel .
Sampling Rate	Not yet supported for RT-LAB data source. Number of samples acquired by second. Use the Probe Control Panel's decimation factor parameter instead.
Number of acquisition	Numbers of acquisition that will be executed.
Synchronization	Activate or deactivate the dynamic trigger of the RT-LAB acquisition group. Signals that trigger that acquisition must be selected using the Probe Control Panel .
Operation Sequence	Not yet available with RT-LAB data source.

Example

The following figure illustrates the behaviour of the acquisition when the following parameters are specified and trigger is enabled:

- Trigger and synchronisation enabled.
- Trigger activated by the model after 2 seconds of simulation.
- Start time = 1 second.
- Time length = 3 seconds.
- Sampling rate = 1000 samples by second.



Open ScopeView without RT-LAB

ScopeView could be executed without running RT-LAB. The next steps explain how to run ScopeView from the file system. The ScopeView executable is available under the installation folder of RT-LAB (RT-LAB/ScopeView).

The RT-LAB default installation folder is: C:\OPAL-RT\RT-LABXX.X, where XX.X is the version of RT-LAB. It could vary based on the installation path. To get the path of RT-LAB, consult the environment variable RTLAR_ROOT or type set RTLAR_ROOT from the windows command line (cmd.exe).

To open ScopeView from the file system:

1. Open Windows Explorer (Win+E)
2. Go to the RT-LAB installation folder (type %RTLAR_ROOT% into address bar)
3. Change directory to ScopeView
4. Double click on ScopeView.exe

Hint: A shortcut of ScopeView could be created and copied on the windows desktop.

Embedding Simulation

Introduction

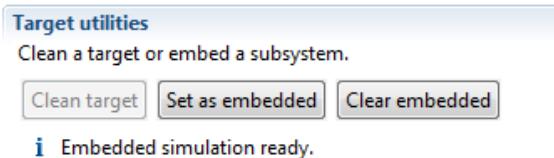
An embedded simulation is a simulation that always runs on a dedicated target. It is part of the target and is automatically loaded and executed when the target is powered on, i.e. it doesn't require the user interface on the host computer to start the simulation. It is properly suited for stand-alone systems such as PC/104 or any embedded computer. As any simulation, it can use any hardware input/output device or asynchronous process to integrate external physical components.

Embedded simulations may be controlled like any simulation using any host computer that runs RT-LAB user interface, even if the simulation was started by another user. See [Enabling Embedded Simulation](#) and [Connecting and Interacting with Embedded Simulation](#) sections. However, all users should share the same RT-LAB project (or a copy) that contains the project's files such as models and configurations files. Note however that any user may disable an embedded simulation at the next target power up without the original project. See [Disabling Embedded Simulation](#).

Enabling Embedded Simulation

Here are the steps to create an embedded simulation.

- Select the project's model you want to embed from the [Project Explorer](#) view.
- Load and execute your model. During this step, RT-LAB will automatically transfer to the target all the required files to embed a simulation. It also authenticates and validates that the simulation is correctly running on the target.
- Open the [Model Editor](#) and select the [Assignment Page](#).
- Click the **Set as Embedded** button in the [Target Utilities](#) section. At this point, your model is embedded and your simulation will start automatically the next time your target is powered up.



- Continue to work with your project and model. If required it is possible to stop and reset your simulation and change parameters and configurations. Note, however, that when a simulation is restarted after a power up of the target, it resets:
 - all model parameters to the original values set when the model was compiled and
 - all model properties and configurations to the values set the last time the the model was loaded through the user interface. As mentionned above, configurations and properties files are transferred to the target during load.
- Close your project to disconnect it from the embedded simulation. You can reconnect the project to the current embedded simulation as described in [Connecting and Interacting with Embedded Simulation](#). You can also open a new instance of the project, for example to embed it on another target. Note that in this case, if you modify the model that is already embedded, you will not be able to reconnect to the previously embedded simulation anymore.

Viewing and Detecting Embedded Simulation

RT-LAB displays embedded simulations in your **Project Explorer**. Normally, you should see them on any target running an embedded simulation. If you expand your target, you will see the project and model running on the target.

RT-LAB also displays embedded simulations directly on the opened or closed projects as in the following figures:

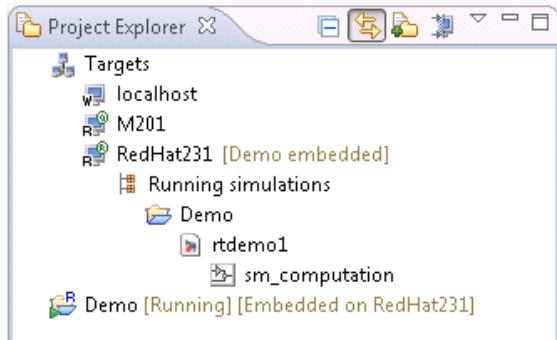


Figure 24: Embedded simulation in Project Explorer

However, the RT-LAB user interface does not automatically detect that a simulation is running on a target after a target's reboot or if a simulation was started by another user. In these particular cases, you could force the user interface to refresh its list of embedded simulations by refreshing one or more targets using the interface:

- Select one or more targets from the **Project Explorer** view.
- Right-click on the items and select **Refresh**. At this point, RT-LAB will refresh the target items and will display embedded simulation on the target if running on the target.

Connecting and Interacting with Embedded Simulation

As explained in the previous section, RT-LAB displays embedded simulations in the **Project Explorer** view as children of targets. To start interacting with an embedded simulation, right click on one of its items (project, model or subsystem) and select **Connect**. This will open the **Connect to Embedded Simulation Wizard** and let you choose the project you want to connect to this embedded simulation.

Once connected, the project can be used to interact with the model like any 'standard' project.

Again, if the embedded simulation was started by another user, your workspace project must contain an exact copy of the embedded model. This is important because RT-LAB will only verify the path and name of the model, not its content.

If you do not have a shared project or a copy of the original project, you should not interact with the embedded simulation. In this case, the only safe action is to disable the embedded mode on the target so the model will not start the next time the target is powered on.

Disabling Embedded Simulation

The preferred approach to disabling the embedded mode is to connect to the embedded simulation as described in the previous section and then:

- Open the **Model Editor** and select the **Assignment Page**.

-
- Click the **Clear embedded** button from the [Target Utilities](#) section.
 - If required, reset your model to stop the current simulation. Clearing and disabling an embedded simulation only removes the settings on the target that start automatically the simulation at power up.

The second approach is based on the target items from the [Project Explorer](#) view. This method is suggested when you do not share the project (or do not have a copy of the project):

- Select the target that runs the embedded simulation in the [Project Explorer](#) view.
- Right click on the target and select **Execute > Remove embedded mode**.
- If a project is currently opened and connected to this embedded simulation, the user interface will prompt you to reset the model.
- Otherwise, if no project is connected to this embedded simulation, it will prompt you to reboot the target.

Alternatively, embedded simulations can be disabled using the Telnet [Terminal View](#) and shell commands:

- Select the target that display embedded simulation in the [Project Explorer](#) view .
- Click the item **Tools > Telnet** to open the Telnet [Terminal View](#).
- Login as root user and enter the following command:
`>> rm -f /usr/opalrt/local/OpalStartup`.

Installing Embedded Target

Previous versions of RT-LAB (8.4.x and lower), require the installation of a special archive file, named *rt-lab_e.tgz*, on the target to support embedded simulations. This step is now automatically performed during the general target installation of RT-LAB, so no extra step is required. Moreover, the embedded mode is now available for both QNX and RedHat platforms.

Known Limitations

Embedded simulation has the following limitations:

- Only one model per project can be embedded on a target.
- Only one model per target can run in embedded mode.
- Embedded simulation is limited to models that have only one real-time subsystem (SM Master). This means that models that contain multiple subsystems could not be embedded on a target.
- Users that did not start an embedded simulation need to share (or have a copy of) the original project to connect to the embedded simulation.
- Embedded mode is not supported on RT-LAB 10.0 and 10.1.

Taking a Snapshot

Introduction

The Snapshot feature has two main uses:

- to restore a set of values in your simulation
- to generate a report on the state of the simulation at a given time.

A **Snapshot** contains the values of all the model's parameters, signals and states at a given time of the real-time simulation. It also contains information about the solvers and data required by s-functions.

Users can take a snapshot of a model while the real-time simulation is running, and then restore it later if desired. For example, if a simulation must run for two days to achieve a given state, a snapshot of the model can be taken after it has run for two days. When the simulation is run again, the snapshot's values can be restored, saving two days of simulation. Also note that to restore a snapshot the real-time simulation must be paused.

To take or restore a snapshot; use the corresponding item in the **Simulation menu** or **Main Toolbar**.

A **Snapshot** block may also be placed in the model, helping the user to automatically take snapshots based on the behaviour of the model. See the Snapshot block in the **Block Library Reference Guide** for more help.

Registering a User S-Function for Snapshot

Note that the following paragraphs are designed for advanced users developing S-Functions with Simulink. If you are not familiar with S-Function, skip this section.

To save data contained in an S-Function when a snapshot is taken, you must register your S-Function in RT-LAB. To do this, use the **OpalSnapshotRegister** function in your s-function. This function enables you to register data so it can be saved when a snapshot is taken, and reloaded when the snapshot is restored.

Follow these steps to register your s-function

1. Include the **model_main.h** header file in your s-function source code:

```
#include "model_main.h"
```

This file is located in the **<RT-LAB directory>\common\include_target** directory on a WINDOWS OS environment, or in **/usr/opalrt/common/include** directory on a QNX or RedHat environment.

2. Define a structure to be copied with the snapshot . For example:

```
def struct
{
    real_T value;
    unsigned int count;
    char string[50];
}LocalData;
```

Once your S-Function is registered, every snapshot taken will contain a raw copy (memcpy is used) of your data.

Note that the structure definition must not contain pointers since snapshot will not copy the area referred by your data pointer. In this case, when a snapshot is restored the pointers will be reset to their previous value and you will get an invalid pointer that may corrupt your s-function and the real-time simulation. This problem is caused by the fact that the memory allocation that was performed for the current simulation may be different from the memory allocation performed when the snapshot was taken.

3. Initialize and register your structure in the mdlInitializeCondition function :

```
// Initialize your data
LocalData *data;
Data = (LocalData *) malloc(sizeof(LocalData));

// Register your data pointer to RT-LAB
OpalSnapshotRegister((void *)data, sizeof(LocalData), ssGetPath(S));
```

First argument: void pointer to the memory block to copy.
Second argument: size (in bytes) of the memory block to copy.
Third argument: path of the S-function (available from the **ssGetPath** macro).

4. Free your data structure in mdlTerminate function:

```
if (data != NULL) {
    free(data);
}
```

5. Before compiling your s-function as a shared library (.mexw32 file), add the **model_main.h** include path to your compilation command. Example:

```
cc [...] -I<RT-LAB directory>\common\include_target [...]
```

Working with perspectives

Perspectives define the initial set and layout of views in the Workbench window. They provide a set of functionality aimed at accomplishing a specific type of task or working with specific types of resources.

See the Related tasks links for more details.

Related concepts

[Perspectives](#)

[Views](#)

[Fast views](#)

[Detached views](#)

Switching between perspectives

Open perspectives are represented by icons on the perspective bar. When you have more than one perspective open, you can switch between them by clicking the icons on the shortcut bar.

Related concepts

[Perspectives](#)

Specifying the default perspective

The default perspective is indicated in the **Select Perspective** dialog (accessible via the **Window > Open Perspective > Other...** menu). The pre-defined default perspective is indicated by the word **default** in brackets following the perspective name, for example, **Edition (default)**.

To change the default perspective:

- Open the **General > Perspectives** preference page.
- Select the perspective that you want to define as the default from the list of available perspectives, and click **Make Default**. The default indicator moves to the perspective that you selected.
- Click **OK**.

Related concepts

[Workbench](#)
[Perspectives](#)

Opening perspectives

Perspectives provide combinations of views and editors that are suited to performing a particular set of tasks. For example, you would normally open the Debug perspective to debug a Python script.

To open a new perspective:

- Click the **Open Perspective** button on the shortcut bar on the left side of the Workbench window. (This provides the same function as the **Window > Open Perspective** menu on the menu bar.)
- To see a complete list of perspectives, select **Other...** from the drop-down menu.
- Select the perspective that you want to open.

When the perspective opens, the title bar of the window it is in changes to display the name of the perspective. In addition, an icon is added to the shortcut bar, allowing you to quickly switch back to that perspective from other perspectives in the same window.

By default, a perspective will open in the same window. If you would rather it opened in a new window, change the setting in the **General > Perspectives** preference page.

Related concepts

[Perspectives](#)

Changing where perspectives open

You can change the default behavior for how perspectives are opened in the Workbench.

- Open the **General > Perspectives** preference page.
- Select either **In the same window** or **In a new window** from the **Open a new perspective** group.
- Click **OK**.

Related concepts

[Perspectives](#)

Configuring perspectives

In addition to configuring the layout of your perspective you can also control several other key aspects of a perspective. These include:

- The options available on the **File > New** submenu.
- The options available on the **Window > Open Perspective** submenu.
- The options available on the **Window > Show View** submenu.
- Action sets (buttons and menu options) that show up on the toolbar and menu bar.

To configure a perspective:

- Switch to the perspective that you want to configure.
- Select **Window > Customize Perspective....**
- Expand the item that you want to customize.
- Use the checkboxes to select which elements you want to see on drop-down menus in the selected perspective. Items you do not select will still be accessible by clicking the **Other** menu option.
- Click **OK**.

Related concepts

[Workbench](#)
[Perspectives](#)

Saving a user defined perspective

If you have modified a perspective by adding, deleting, or moving (docking) views, you can save your changes for future use.

- Switch to the perspective that you want to save.
- Click **Window > Save Perspective As....**
- Type a new name for the perspective into the **Name** field.
- Click **OK**.

The name of the new perspective is added to the **Window > Open Perspective** menu.

Related concepts

[Perspectives](#)

[Views](#)

Deleting a user defined perspective

You can delete perspectives that you defined yourself, but not those that are delivered with the Workbench.

- Open the **General > Perspectives** preference page.
- From the **Available perspectives** list, select the one that you want to delete and click **Delete**.
- Click **OK**.

Related concepts

[Perspectives](#)

Resetting perspectives

To restore a perspective to its original layout:

- Open the **General > Perspectives** preference page.
- From the **Available perspectives** list, select the perspective you want to restore.
- Click **Reset**.
- Click **OK**.

Related concepts

[Perspectives](#)

Working with views and editors

Views and editors are the main visual entities which appear in the Workbench. In any given perspective there is a single editor area, which can contain multiple editors, and a number of surrounding views which provide context.

The Workbench provides a number of operations for working with views and editors.

Related concepts

- [Views](#)
- [Editors](#)
- [Fast views](#)
- [Detached views](#)
- [Perspectives](#)

Opening views

Perspectives offer pre-defined combinations of views and editors. To open a view that is not included in the current perspective, select **Window > Show View** from the main menu bar.

You can create fast views to provide quick access to views that you use often.

After adding a view to the current perspective, you may wish to save your new layout by clicking **Window > Save Perspective As....**

Related concepts

[Views](#)

[Fast views](#)

[Detached views](#)

[Perspectives](#)

Moving and docking views

To change the location of a view in the current perspective:

- Drag the view by its title bar. Do not release the left mouse button yet.
- As you move the view around the Workbench, the mouse pointer changes to one of the drop cursors shown in the table below. The drop cursor indicates where the view will be docked if you release the left mouse button. To see the drop cursor change, drag the view over the left, right, top, or bottom border of another view or editor. You may also drag the view outside of the Workbench area to turn it into a "Detached" view.
- When the view is in the location that you want, relative to the view or editor area underneath the drop cursor, release the left mouse button.
- (Optional) If you want to save your changes, select **Window > Save Perspective As...** from the main menu bar.
- Note that a group of stacked views can be dragged using the empty space to the right of the view tabs.

You can also move a view by using the pop-up menu for the view. (Left-click on the icon at the left end of the view's title bar, or right-click anywhere else in the view's title bar). As well as moving the view this menu will provide shortcut options for turning a view into either a "Fast" or "Detached" view.

Drop cursor	Where the view will be moved to
↑	Dock above The view is docked above the view underneath the cursor.
↓	Dock below The view is docked below the view underneath the cursor.
→	Dock to the right The view is docked to the right of the view underneath the cursor.
←	Dock to the left The view is docked to the left of the view underneath the cursor.
📁	Stack The view is docked as a Tab in the same pane as the view underneath the cursor.
☒	Detached The view is detached from the Workbench window and is shown in its own separate window.
🚫	Restricted You cannot dock the view in this area.

Related concepts

- [Views](#)
- [Fast views](#)
- [Detached views](#)
- [Perspectives](#)

Rearranging tabbed views

In addition to dragging and dropping (docking) views inside the Workbench, you can rearrange the order of views within a tabbed notebook.

- Click on the tab of the view that you want to move and drag it to where you want it. A stack symbol appears as you drag the view across other view tabs.
- Release the mouse button when you have the view tab in the desired location. The view that you selected is now moved.

Related concepts

[Views](#)

Creating fast views

Fast views are hidden views that can be quickly opened and closed. They work like other views except they do not take up space in your Workbench window.

To create a fast view:

- Click the title bar of the view that you want. Hold the mouse button down.
- Drag the view to the Fast View bar and release the mouse button. By default the shortcut bar is located in the lower left corner of the window.

Alternatively, you can click on the button located on the left side of the Fast View bar which will present you with a selection of views. Choosing one of these views will add it to the Fast View bar immediately.

The icon for the view that you dragged now appears on the shortcut bar. You can look at the view by clicking its icon on the shortcut bar. As soon as you click somewhere else outside the view, it is hidden again.

To restore the view to its original location (and remove it from the Fast View bar), toggle the fast view item in the view button's context menu.

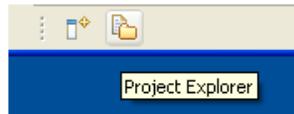
You can also create and restore fast views by selecting **Fast View** from the context menu that opens when you click the icon at the left side of the view's title bar.

Related concepts

- [Views](#)
- [Fast views](#)
- [Perspectives](#)

Working with fast views

If you have converted the Project Explorer to a fast view it will appear in the shortcut bar as shown below.



To work with a fast view proceed as follows.

- In the shortcut bar click on the Project Explorer fast view button.
- Observe the Project Explorer view slides out from the shortcut bar.
- You can use the Project Explorer fast view as you would normally.
- To hide the fast view simply click off of it or click on the Minimize button on the fast view's toolbar

Note: If you open a file from the Project Explorer fast view, the fast view will automatically hide itself to allow you to work with the file.

To convert a fast view that has been maximized back to a regular sized view, select **Restore** from the context menu of the icon in the top left corner of the view. To reposition a fast view, drag the fast view's title bar (or close the fast view and then drag its button from the shortcut bar) and drop it on the workbench like a normal view.

Related concepts

[Views](#)
[Fast views](#)
[Perspectives](#)

Detaching views

Detached views are views that are shown in a separate window with a smaller trim. They work like other views except they are always shown in front of the Workbench window. It is supported on Windows only.

To detach a view:

- If the Workbench window is maximized, resize it so that it does not fill the entire screen.
- Click the title bar of the view that you want to detach. Hold the mouse button down.
- Drag the view to the outside of the Workbench window and release the mouse button.

To restore the view to be shown inside of the Workbench window, drag it into the Workbench window.

You can also detach a view by selecting **Detached** from the context menu that opens when you click the icon at the left side of the view's title bar.

Related concepts

[Views](#)
[Detached views](#)
[Perspectives](#)

Opening files for editing

You can launch an editor for a given file in several ways.

- By right-clicking the file in the Project Explorer view and then selecting **Open** from the pop-up menu.
- By double-clicking the file in one of the navigation views.

All of the above alternatives open the file in the default editor for that type of file. To open it in a different editor, select Open With from the file's pop-up menu.

Related concepts

[Editors](#)

[External editors](#)

[Model Editor](#)

[Project Explorer view](#)

Editing files outside the Workbench

To edit a Workbench resource outside the Workbench:

- Navigate in the file system to the Workbench's installation directory. Go into the workspace directory and open the file that you want to edit with the external editor.
- Edit the file as needed. Save and close it as usual.
- Important: Go back to the Workbench, right-click the edited file in one of the navigation views, and select **Refresh** from the pop-up menu. The Workbench will perform any necessary update operations to process the changes that you made outside the Workbench.

Tip: If you work with external editors regularly, you may want to enable auto-refresh. This can be done by opening the **General > Workspace** preference page, and checking the **Refresh automatically** option. When this option is enabled, any external changes will be automatically discovered by the Workbench. Depending on the platform this may not happen immediately.

Related concepts

[Editors](#)

[External editors](#)

[Project Explorer view](#)

Tiling editors

The Workbench allows you to have multiple files open in multiple editors. Unlike views, editors cannot be dragged outside the Workbench to create new windows. However, you can tile editor sessions within the editor area, in order to view source files side by side.

- With two or more files open in the editor area, select one of the editor tabs.
- Holding down the left mouse button, drag that editor over the left, right, top or bottom border of the editor area. Notice that the mouse pointer changes to a "drop cursor" that indicates where the editor session will be moved when you release the mouse button.
- (Optional) Drag the borders of the editor area or each editor, to resize as desired.

This is a similar operation to moving and docking views inside the Workbench, except that all editor sessions must be contained within the editor area.

Related concepts

[Editors](#)

Maximizing and minimizing elements of the workbench presentation

RT-LAB presentation provides a rich environment consisting of (in its basic form) an Editor Area (containing one or more stacks showing the open editors) surrounded by one or more View Stacks (each containing one or more views). These various parts compete for valuable screen real-estate and correctly managing the amount of screen given to each can greatly enhance your productivity within the IDE.

The two most common mechanisms for managing this issue are 'minimize' (i.e. make me use as little space as possible) and 'maximize' (i.e. give me as much space as you can). The RT-LAB presentation provides a variety of ways to access these operations:

- Using the minimize and maximize buttons provided on a stack's border
- Selecting the 'Minimize' or 'Maximize' item on the context (right-click) menu for a stack
- Double-clicking on a stack
- Using 'Ctrl + M': this is a key binding for a command that will toggle the currently active part between its 'maximized' and its 'restored' (i.e. normal) states.

Maximize

It is desirable at times to focus your attention on one particular part to the exclusion of the others. The most popular candidate for this is, of course, maximizing the editor area in order to make as much of the display available for editing as possible (but there are workflows where it would make sense to focus on a view as well).

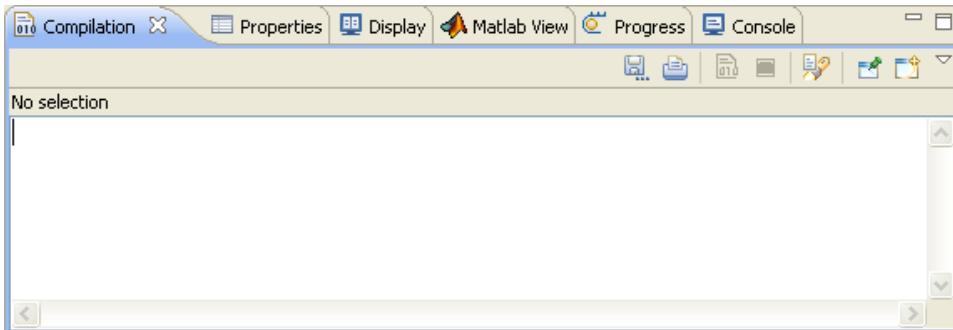
The default presentation implements the maximize behavior by minimizing all stacks except the one being maximized. This allows the maximized stack to completely occupy the main presentation while still allowing access any open views in your perspective by using the icons in their Trim Stack (the area around the edges of the window is called the 'trim').

The behavior for managing the editor maximization operate on the complete Editor Area (rather than simply maximizing the particular Editor Stack. This allows for 'compare' workflows which require the ability to see both files in a split editor area at the same time.

Minimize

Another way to optimize the use of the screen area is to directly minimize stacks that are of no current interest. The default presentation minimizing a stack will cause it to be moved into the trim area at the edges of the workbench window, creating a Trim Stack. View Stacks get minimized into a trim representation that contains the icons for each view in the stack.

This view stack



becomes this Trim Stack when minimized



The minimize behavior for the Editor Area is somewhat different; minimizing the Editor Area results in a trim stack containing only a placeholder icon representing the entire editor area rather than icons for each open editor (since in most cases all the icons would be the same, making them essentially useless).

The editor area



becomes this Trim Stack when minimized



If your particular workflow is such that you need to have more than one element (i.e. having the Editor Area and a View Stack in the presentation at the same time) you can still gain additional screen space by minimizing the stacks that aren't of current interest. This will remove them from the main presentation and place them on the outer edge of the workbench window as Trim Stacks, allowing more space for the remaining stacks in the presentation.

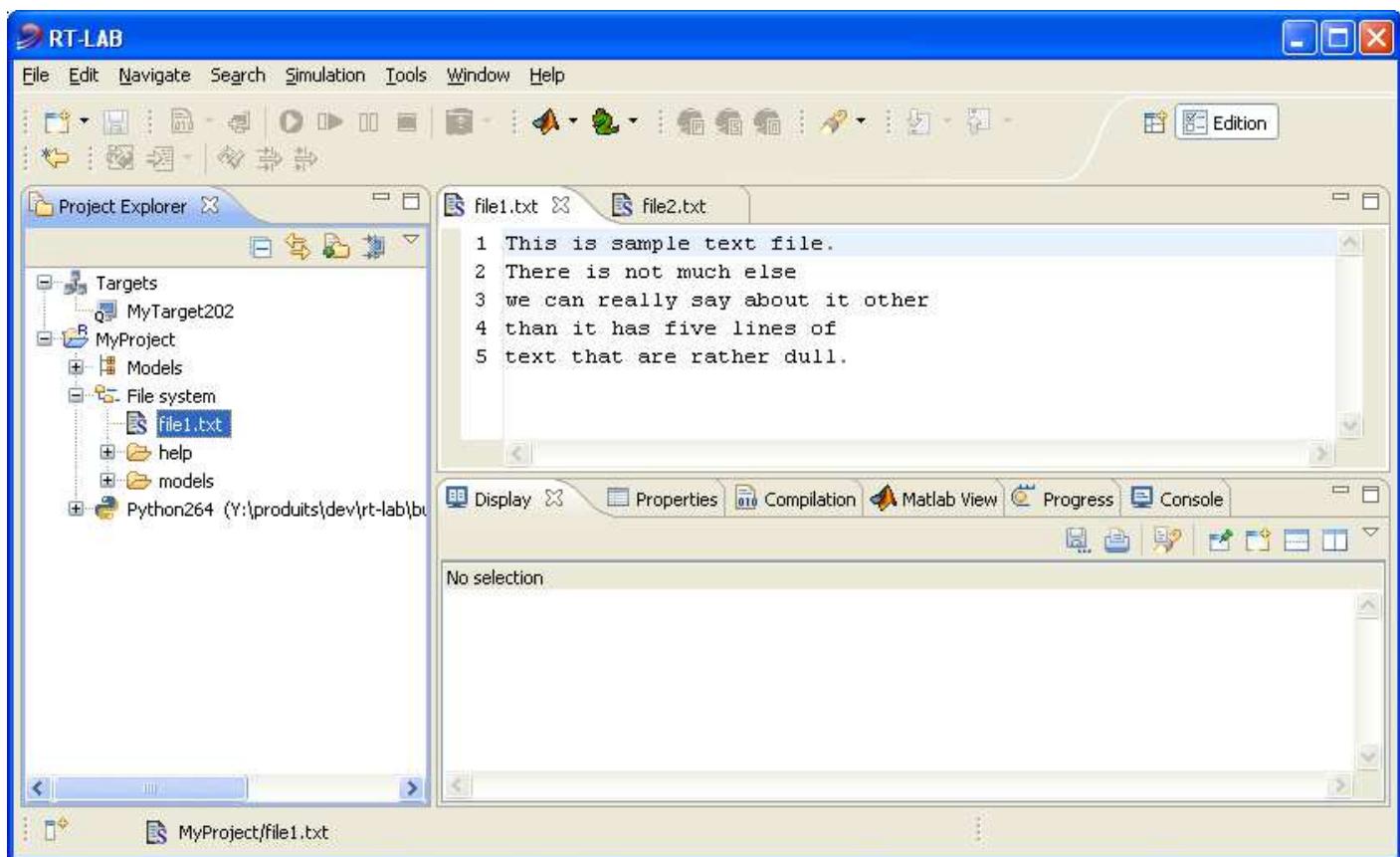
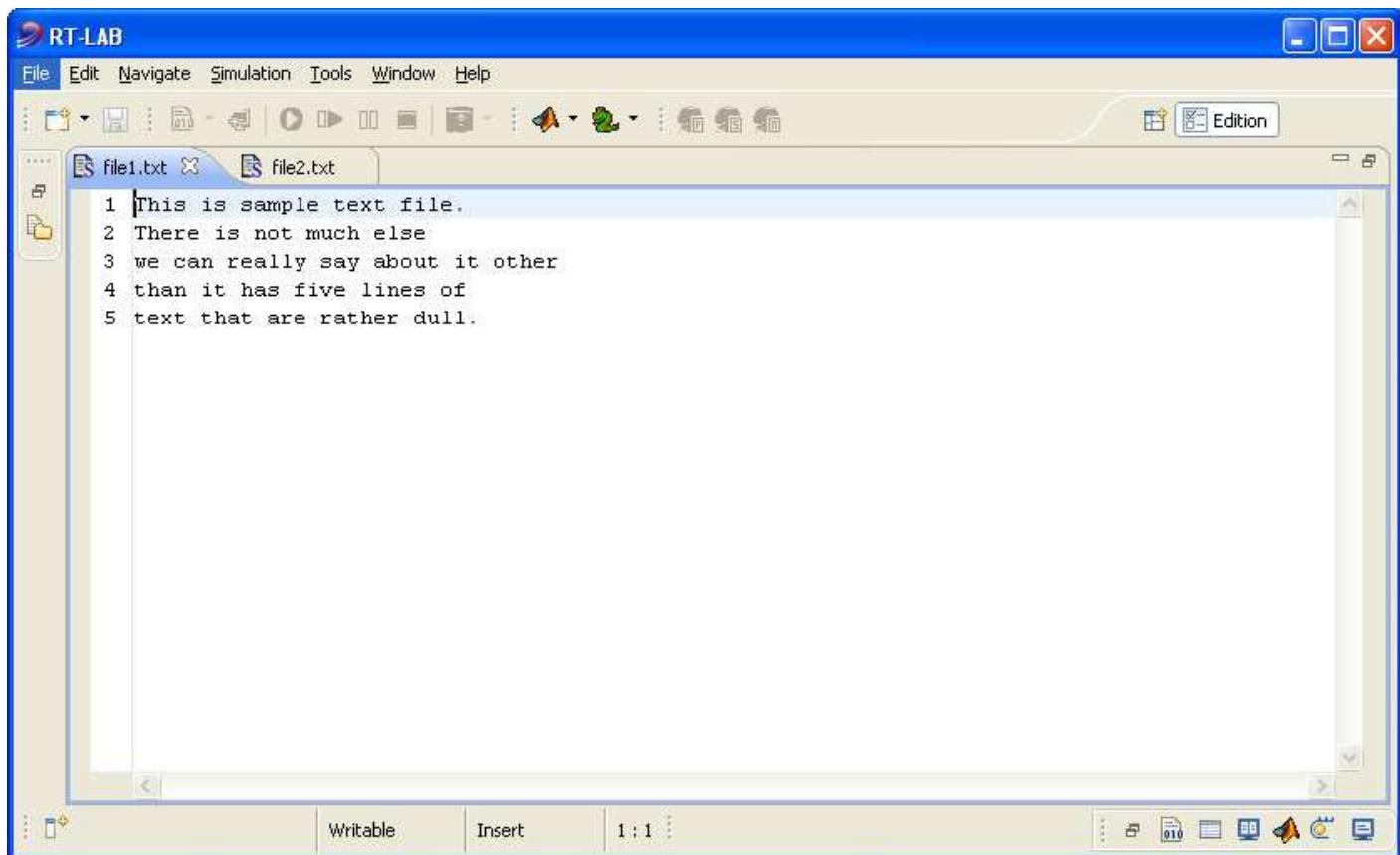
Note: There are two ways to end up with a stack in the trim:

- Directly minimizing the stack
- As the result of another stack being maximized

Depending on how the Trim Stack was created its behavior is different; when un-maximizing only those trim stacks that were created during the initial maximize will be restored to the main presentation while stacks that were independently minimized stay that way.

Tip: This difference is important in that it allows you fine grained control over the presentation. While using maximize is a one-click operation it's an 'all or nothing' paradigm (i.e. no other stack is allowed to share the presentation with a maximized stack). While adequate for most tasks you may find yourself wanting to have the presentation show more than stack. In these scenarios don't maximize; minimize all the other stacks except the ones you want in the presentation. Once you have it set up you can still subsequently maximize the editor area but the un-maximize will only restore the particular stack(s) that were sharing the presentation, not the ones you've explicitly minimized.

Normal Presentation

**Editor Area Maximized**

Related concepts

[Editors](#)

Customizing the Workbench

Many aspects of the appearance and behavior of the Workbench can be customized to suit your individual needs. For example, you can:

- Rearrange where items appear in the main toolbar.
- Change the key bindings used by editors.
- Change the fonts and colors which are used.

Related concepts

[Views](#)

[Editors](#)

[Workbench](#)

Customizing the Welcome

The welcome appearance can be customized via the "customize page" button above the welcome page. This opens a customize dialog which allows you to select one of the pre-defined themes, which affects the overall look of the welcome. You can also select which pages will be displayed, and the visibility, layout, and priority of the items within each page.

Related concepts

[Welcome](#)

Workspace Switching

The current workspace for RT-LAB can be switched by using the File->Switch Workspace command.

The Switch Workspace menu item will open the switch workspace dialog. The dialog will allow you to browse for or manually enter a new workspace location. The combo will also allow you to select your previously selected workspaces.

Settings Transfers

When you switch your workspace you can select settings than will be transferred to the new workspace.

RT-LAB supplies transfers for:

- Workspace Layout: Opened views, thier size, and selected perspectives.
- Working Sets: The user defined working sets.
-

Rearranging the main toolbar

You can rearrange sections of the main toolbar. Toolbar sections are divided by a thin vertical line.

- Make sure the toolbar is unlocked. The toolbar is unlocked if it has thick vertical bars next to the thin vertical toolbar dividers.

If it is locked, unlock the toolbar by right clicking the toolbar and selecting the **Lock the Toolbars** menu item.

- Grab the section of the toolbar you want to rearrange by moving the mouse over the thick vertical line on the left side of the desired segment. The mouse cursor changes its shape to indicate that you can click to move the toolbar section.
- Click and hold the left mouse button to grab the toolbar section.
- Move the section left and right or up and down. Release the mouse button to place it in the new location.
- To prevent accidental changes to the toolbar lock it again by right clicking the toolbar and selecting the **Lock the Toolbars** menu item.

Related concepts

[Toolbars](#)

Changing the key bindings

The function of the keyboard can be extensively customized in RT-LAB.

Use the **General > Keys** preference page to assign key sequences to many of the commands in RT-LAB.

Changing font and colors

By default, the Workbench uses the fonts and colors provided by the operating system. However, there are a number of ways that this behavior can be customized.

Fonts

The Workbench lets you directly configure the following fonts:

Banner Font

Used in PDE editors, welcome pages and in the title area of many wizards. For instance the New Project wizard uses this font for the top title.

Dialog Font

Used for widgets in dialogs.

Header Font

Used as a section heading. For instance the Welcome page for the RT-LAB Platform uses this font for the top title.

Text Font

Used in text editors.

Ignored Resource Font

Used to display resources that are ignored from CVS.

Outgoing Change Font

Used to display outgoing changes in CVS.

Detail Pane Text Font (defaults to text font)

Used by the debug console.

Properties File Editor Text Font (defaults to text font)

Used by Properties File editors.

Compare Text Font (defaults to text font)

Used by textual compare/merge tools.

Part Title Font (defaults to properties file editor text font)

Used for view and editor titles. Note: It is recommended that this font not be bold or italic because the workbench will use bold and italic versions of this font to display progress.

View Message Font (defaults to properties file editor text font)

Used for messages in the view title bar (if present).

To change these fonts:

- Open the **General > Appearance > Colors and Fonts** preference page.
- Select the font you want to change.
- Click **Change**.
- Use the dialog which opens to select a font.
- Click **OK**.

Note: You can also click **Use System Font** to set the font to a reasonable value chosen by the operating system. For example, on Windows this will use the font selected in the Display Properties control panel.

Plug-ins that use other fonts may also provide preference entries to allow them to be customized.

In addition to the above, some text is always displayed in the system font. For example, the navigator tree always does this. To change the font used in these areas, you can use the configuration tools provided by the operating system (for example, the Display Properties control panel on Windows).

Colors

To set the colors used by the Workbench to display error text and hyperlink text:

- Open the **General > Appearance > Colors and Fonts** preference page.
- Select the color you want to change in the tree view and click the color bar on the right.
- Use the dialog which opens to select a color.
- Click **OK**.

Plug-ins that use other colors may also provide preference entries to allow them to be customized. For example, the searching support provides a preference for controlling the color used to display potential matches (see the **Foreground color for potential matches** item on the **General > Search** preference page).

In general, the Workbench uses the colors that are chosen by the operating system. To change these colors you can use the configuration tools provided by the system (for example, the Display Properties control panel on Windows).

Changing the placement of the tabs

You can change the placement of the tabs. The tabs for stacked views or editors can appear at the top or bottom of the area which contains them.

- Open the **General > Appearance** preference page.
- Select from the choices displayed in the **Editor tab positions** group or **View tab positions** group to control whether you want the tabs at the top or the bottom.
- Click **Apply** or **OK**.
- The tabs will immediately move to their new locations.

Related concepts

[Views](#)
[Editors](#)

Legal

Some content of the present documentation is part of the official Eclipse documentation. This content is distributed under the Eclipse Public License available at <http://www.eclipse.org/legal/epl-v10.html>.

All other content that is related to RT-LAB or to other OPAL-RT Software is distributed under OPAL-RT copyright: © 2007 Opal-RT Technologies Inc. All rights reserved for all countries.

