

**Real-Time Interface (RTI and RTI-MP)**

# **Implementation Reference**

**Release 6.0 – November 2007**



## How to Contact dSPACE

Mail:	dSPACE GmbH Technologiepark 25 33100 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 66529
E-mail:	info@dspace.de
Web:	<a href="http://www.dspace.com">http://www.dspace.com</a>

## How to Contact dSPACE Support

There are different ways to contact dSPACE Support:

- Visit our Web site at <http://www.dspace.com/goto?support>
- Send an e-mail or phone:
  - General Technical Support:  
[support@dspace.de](mailto:support@dspace.de)  
+49 5251 1638-941
- Use the dSPACE Support Wizard:
  - On your dSPACE DVD at \Diag\Tools\dSPACESupportWizard.exe
  - Via **Start – Programs – dSPACE Tools** (after installation of the dSPACE software)
  - At <http://www.dspace.com/goto?supportwizard>

You can always find the latest version of the dSPACE Support Wizard here.

dSPACE recommends that you use the dSPACE Support Wizard to contact dSPACE Support.

## Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit <http://www.dspace.com/goto?support> for software updates and patches.

## Important Notice

This document contains proprietary information that is protected by copyright. All rights are reserved. Neither the documentation nor software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© Copyright 2007 by:  
dSPACE GmbH  
Technologiepark 25  
33100 Paderborn  
Germany

This publication and the contents hereof are subject to change without notice.

AutomationDesk, CalDesk, ConfigurationDesk, ControlDesk and TargetLink are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

# Contents

<b>About This Reference</b>	7
<i>Document Symbols and Conventions</i> .....	9
<i>Accessing Online Help and PDF Files</i> .....	11
<i>Related Documents</i> .....	12
 <b>RTI Dialog Reference</b>	15
<i>Model Configuration Parameters Dialogs</i> .....	16
<i>Solver Dialog (Model Configuration Parameters Dialogs)</i> .....	18
<i>Optimization Dialog (Model Configuration Parameters</i> <i>Dialogs)</i> .....	21
<i>Diagnostics Dialog (Model Configuration Parameters Dialogs)</i> .....	23
<i>Hardware Implementation Dialog (Model Configuration</i> <i>Parameters Dialogs)</i> .....	25
<i>Real-Time Workshop Dialog (Model Configuration Parameters</i> <i>Dialogs)</i> .....	27
<i>Model Parameters Configuration Dialog</i> .....	37
<i>RTI Task Configuration Dialog</i> .....	38
<i>External Mode Control Panel</i> .....	40
<i>Signal Properties Dialog</i> .....	42
 <b>RTI-MP Dialog Reference</b>	43
<i>Communication Channels Display</i> .....	44
<i>Change Connection Dialog</i> .....	44
<i>Multiprocessor Setup Dialog</i> .....	45
<i>Main Page (Multiprocessor Setup Dialog)</i> .....	46
<i>CPU's Page (Multiprocessor Setup Dialog)</i> .....	49
<i>Advanced Page (Multiprocessor Setup Dialog)</i> .....	54
<i>Documentation Page (Multiprocessor Setup Dialog)</i> .....	56
<i>RTI Task Configuration Dialog (Multiprocessor Setup Dialog)</i> .....	56
<i>CPU Options Dialog</i> .....	58
<i>Build Options Page (CPU Options Dialog)</i> .....	58
<i>Variable Description File Options Page (CPU Options Dialog)</i> .....	61
<i>Rename CPU Dialog</i> .....	64
<i>Multiprocessor Topology Setup Dialog</i> .....	65
<i>Model Configuration Parameters Dialogs</i> .....	67
<i>Optimization Dialog (Model Configuration Parameters</i> <i>Dialogs)</i> .....	69

<i>Diagnostics Dialog (Model Configuration Parameters Dialogs)</i>	73
<i>Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)</i>	75
<i>Model Parameters Configuration Dialog</i>	76
<i>External Mode Control Panel</i>	77
<i>Signal Properties Dialog</i>	78

<b>RTI and RTI-MP Variable Reference</b>	81
<i>currentTime</i>	82
<i>errorNumber</i>	83
<i>finalTime</i>	83
<i>modelStepSize</i>	84
<i>overrunCheckType</i>	84
<i>overrunCount</i>	85
<i>overrunQueueCount</i>	85
<i>overrunQueueMax</i>	86
<i>priority</i>	86
<i>rtiAssertionMode</i>	87
<i>sampleTime</i>	88
<i>simState</i>	88
<i>state</i>	89
<i>taskCallCount</i>	89
<i>turnaroundTime</i>	90

<b>RTI and RTI-MP File Reference</b>	93
<i>File Overview</i>	94
<i>Simulink Files</i>	94
<i>Files Generated by RTI and RTI-MP</i>	94
<i>Files Controlling the Build and Download Process</i>	95
<i>Executable Real-Time Object and System Description Files</i>	97
<i>Variable Access Files for ControlDesk/MLIB/CLIB</i>	98
<i>User-Supplied Files</i>	99
<i>File Details</i>	101
<i>startup.m</i>	101
<i>dsstartup.m</i>	102
<i>dsfinish.m</i>	103
<i>User-Code File (USR.C File)</i>	103
<i>User Makefile (USR.MK File)</i>	105
<i>Linker Command File (LK File)</i>	108
<i>User System Description File (USR.SDF File)</i>	108
<i>Variable Description File (TRC File)</i>	109
<i>Variable Description File Groups</i>	113
<i>Available Variables in the Variable Description File</i>	115
<i>Data Type Specifications</i>	118
<i>User Variable Description File (USR.TRC File)</i>	118

<b>RTI and RTI-MP Command Reference</b>	121
<i>rti_usrtrcmmerge</i> .....	122
<i>rti_sdfmerge</i> .....	122
<i>rti_build</i> .....	122
<i>rti1xxx</i> .....	126
<i>rtiver</i> .....	127
<i>rtimp_blktargetcpunameget</i> .....	127
<i>rtimp_build</i> .....	128
<i>set_rti</i> .....	133
<i>rti_optionset</i> .....	133
<i>rti_optionget</i> .....	135
<i>rti_mdcleanup</i> .....	137
<i>rti_option TaskTransitionCheck</i> .....	138
 <b>General RTI and RTI-MP Libraries</b>	139
TaskLib Block Reference .....	141
<i>Background Block</i> .....	142
<i>Buffered Task Transition (Read) Block</i> .....	143
<i>Buffered Task Transition (Write) Block</i> .....	146
<i>Hardware Interrupt Block</i> .....	148
<i>Function-Call Subsystem Block</i> .....	150
<i>Non-Buffered Task Transition Block</i> .....	151
<i>Software Interrupt Block</i> .....	152
<i>Timer Interrupt Block</i> .....	153
<i>Time-Trigger Set Block</i> .....	156
<i>Time-Triggered Task Block</i> .....	158
<i>Timetable Start Block</i> .....	159
<i>Timetable Task Block</i> .....	160
<i>Timer Task Assignment Block</i> .....	161
<i>Uninterruptable Buffer Block</i> .....	163
Extras Block Reference .....	164
<i>Data Capture Block</i> .....	165
<i>Ramp Generator for Encoder Index Search Block</i> .....	167
<i>simState READ Block</i> .....	169
<i>simState SET Block</i> .....	169
<i>TRC Exclusion Block</i> .....	170
RTI-MP Block Reference .....	172
<i>Default CPU Block</i> .....	173
<i>Interprocessor Communication (IPC) Block</i> .....	174
<i>Interprocessor Interrupt (IPI) Block</i> .....	179
<i>Multiprocessor Setup Block</i> .....	180
 <b>Index</b>	181



---

# About This Reference

---

This reference contains information on the various dialogs, files, options, etc. of Real-Time Interface (RTI and RTI-MP) for dSPACE systems. It also describes RTI-MP blocks and the RTI blocks that are common to all supported platforms. RTI acts as the link between Simulink and the dSPACE hardware. RTI-MP enables you to partition your model and allocate the parts to the different CPUs of a multiprocessor system.



- dSPACE Release 6.0 supports the following MATLAB releases:

- MATLAB R2006a+
- MATLAB R2006b
- MATLAB R2007a+
- MATLAB R2007b

For information on the compatibility of dSPACE Release 6.0 with MATLAB releases that came out later than dSPACE Release 6.0, visit [www.dspace.com/goto?compatibility](http://www.dspace.com/goto?compatibility).

- This document describes Real-Time Interface for all hardware platforms. Not all sections apply to all platforms. You can skip all paragraphs that are not relevant to your platform.
- Where you find terms like `rti<XXXX>` in this document (for example, in path or file names), replace them by the RTI platform support you are using, for example, `rti1005`.
- Where you find terms like `<model>` or `<submodel>` in this document, replace them by the actual name of your model or submodel. For example, if the name of your Simulink model is `smd1103_sl.mdl` and you are asked to edit the `<model>_usr.c` file, you actually have to edit the `smd1103_sl_usr.c` file.

## Where to go from here

Information in this section

<b><i>Document Symbols and Conventions</i></b>	9
<b><i>Accessing Online Help and PDF Files</i></b>	11
<b><i>Related Documents</i></b>	12

Information in other sections

Commands	
<b><i>RTI and RTI-MP Command Reference</i></b>	121
Variables	
<b><i>RTI and RTI-MP Variable Reference</i></b>	81
Files	
<b><i>RTI and RTI-MP File Reference</i></b>	93








Dialogs	
<b><i>RTI Dialog Reference</i></b>	15
<b><i>RTI-MP Dialog Reference</i></b>	43
Block libraries	
<b><i>General RTI and RTI-MP Libraries</i></b>	139
<b><i>TaskLib Block Reference</i></b>	141
<b><i>Extras Block Reference</i></b>	164
<b><i>RTI-MP Block Reference</i></b>	172

## Document Symbols and Conventions

### Symbols

The following symbols may be used in this document.

	Indicates a general hazard that may cause personal injury of any kind if you do not avoid it by following the instructions given.
	Indicates the danger of electric shock which may cause death or serious injury if you do not avoid it by following the instructions given.
	Indicates a hazard that may cause material damage if you do not avoid it by following the instructions given.
	Indicates important information that should be kept in mind, for example, to avoid malfunctions.
	Indicates tips containing useful information to make your work easier.

### Naming conventions

The following abbreviations and formats are used in this document:

**%name%** Names enclosed in percent signs refer to environment variables for file and path names, for example, %DSPACE\_ROOT% specifies the location of your dSPACE installation in the file system.

**< >** Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Examples:


- Where you find terms like `rti<XXXX>` replace them by the RTI platform support you are using, for example, `rti1005`.
- Where you find terms like `<model>` or `<submodel>` in this guide, replace them by the actual name of your model or submodel. For example, if the name of your Simulink model is `smd1103_sl.mdl` and you are asked to edit the `<model>_usr.c` file, you actually have to edit the `smd1103_sl_usr.c` file.


**RTI block name conventions** All I/O blocks have default names based on dSPACE's board naming conventions:

- A block name always starts with the board name.
- A short description of its functionality is added.
- Most RTI block names also have a suffix.

Suffix	Meaning
B	Board number (for modular systems)
M	Module number (for MicroAutoBox)
C	Channel number
G	Group number
CON	Converter number
BL	Channel block
P	Port number
I	Interrupt number

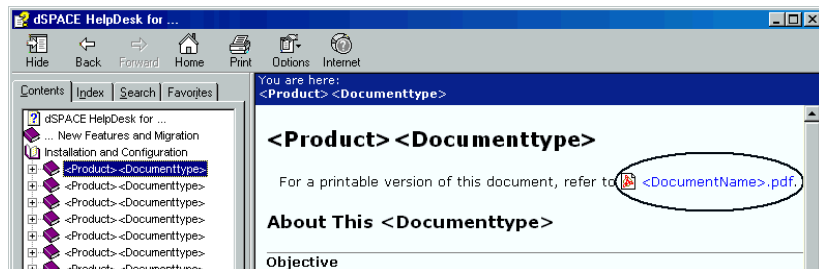
A suffix is followed by the appropriate number. For example, *DS2201IN\_B2\_C14* represents a digital input block located on a DS2201 board. The suffix indicates board number 2 and channel number 14 of the block. For more general block naming, the numbers are replaced by variables (for example, *DS2201IN\_Bx\_Cy*).

 Precedes the document title in a link that refers to another document.

 Indicates that a link refers to another document, which is available in dSPACE HelpDesk.

## Accessing Online Help and PDF Files

<b>Objective</b>	After you install your dSPACE software, the documentation for the installed products is available as online help and Adobe® PDF files.
<b>Online help</b>	<p>You can access the online help – dSPACE HelpDesk – as follows:</p> <p><b>Windows Start menu</b> Click <b>Start – Programs – dSPACE Tools – dSPACE HelpDesk</b>.</p> <p><b>Context-sensitive</b> Press the <b>F1</b> key or click the <b>Help</b> button in the dSPACE software.</p> <p><b>Local installation on your host PC</b> Double-click the dSPACEHelpDesk.chm file in %DSPACE_ROOT%\Doc\Online.</p>
<b>PDF Files</b>	<p>You can access the PDF files as follows:</p> <p><b>dSPACE HelpDesk</b> Click the PDF link at the beginning of a document:</p>



**Local installation on your host PC** Double-click the PDF file in %DSPACE\_ROOT%\Doc\Print.

## Related Documents

Below is a list of documents that you are also recommended to read when working with RTI and RTI-MP.

Information in other documents

RTI and RTI-MP

***RTI and RTI-MP Implementation Guide***

Gives detailed information and instructions on using Real-Time Interface (RTI and RTI-MP) to implement your real-time models.

Processor and controller boards

***DS1005 Implementation Documents***

***DS1006 Implementation Documents***

***DS1103 Implementation Documents***

***DS1104 Implementation Documents***

***MicroAutoBox Implementation Documents***

Connectable I/O boards

***I/O Board Implementation Documents***

Provides detailed information on the features, RTI, RTLib and hardware of the dSPACE I/O boards that can be connected to a dSPACE processor board.

ControlDesk Standard

***ControlDesk Experiment Guide***

Introduces you to the experiment features provided by ControlDesk Standard.

***ControlDesk Automation Guide***

Shows you how to automate the features provided by ControlDesk Standard.

***ControlDesk Reference***

***ControlDesk Instrument Reference***

***ControlDesk Automation Reference***

AutomationDesk

***AutomationDesk Guide***

Introduces you to the automation features provided by AutomationDesk. It provides detailed information on the GUI components, basic concepts, and instructions using AutomationDesk and AutomationDesk Automation Server.

***AutomationDesk Tutorial***

Has several lessons that guide you through using AutomationDesk. It shows you concrete procedures by means of a demo project.

***AutomationDesk Reference***

Provides detailed information on the menus, context menus, and dialogs contained in AutomationDesk.

***AutomationDesk Library Reference***

Provides detailed information on the libraries supported by AutomationDesk.

***Interface libraries******MLIB/MTRACE MATLAB-dSPACE Interface Libraries***

Contains detailed reference information and instructions on the experiment features of MLIB/MTRACE.

***CLIB C Interface Library***

Contains detailed reference information on the C Interface Library CLIB, which contains C functions to access the dSPACE processor and controller boards from the host PC.



# RTI Dialog Reference

The various dialogs of Simulink and RTI provide many options to customize the behavior of Simulink and code generation for the dSPACE real-time platforms.

The following dialogs are relevant when you use RTI:

Purpose	Refer to
Model configuration and simulation parameters	
To specify simulation parameters for single-processor models.	<i>Model Configuration Parameters Dialogs</i> on page 16
To make individual parameters tunable if the Inline parameters option is selected.	<i>Model Parameters Configuration Dialog</i> on page 37
To specify priorities for the tasks of a model.	<i>RTI Task Configuration Dialog</i> on page 38
External simulation	
To control an external simulation.	<i>External Mode Control Panel</i> on page 40
Signal properties	
To make a signal displayable even if certain code optimization options are selected.	<i>Signal Properties Dialog</i> on page 42
Related Topics	
The dialogs described here relate to various files.	<i>RTI and RTI-MP File Reference</i> on page 93
For general information on Simulink's dialogs, refer to the corresponding documentation provided by The MathWorks.	

## Model Configuration Parameters Dialogs

Simulink provides different types of configuration sets. The following configuration sets can be distinguished:

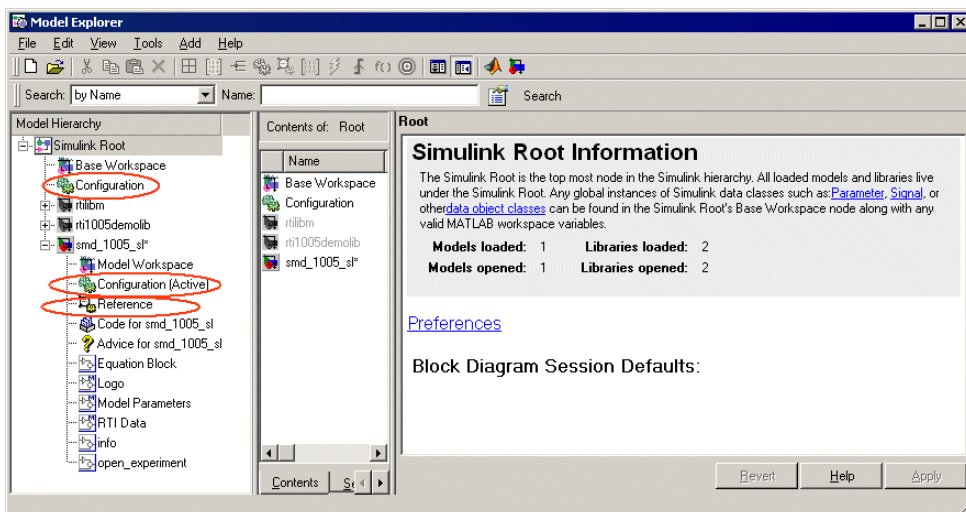
- The Configuration Preferences set
- Model-specific configuration sets
- Referenced configuration sets

**Configuration Preferences set** In the Configuration Preferences set, you can specify default values for all model properties, which are transferred to the model's configuration set when you create a new model.

**Model-specific configuration sets** Your model can have several configuration sets. You can activate a specific configuration set for a simulation or code generation.

**Referenced configuration sets** Your model can have several configuration references that link to configuration sets in the MATLAB base workspace. The referenced configuration sets in the MATLAB base workspace contain the simulation parameters which are used for your model. Referenced configuration sets can be linked by any number of models. I.e., changing the value of any parameter in the configuration set changes it for every model that references this configuration set.

The following illustration shows the different types of configuration sets in the Model Explorer:





**Access**

You can access the **Configuration Preferences** via the **File – Preferences** menu command of the MATLAB Desktop. Then expand the Simulink node in the Preferences tree, and click Launch model explorer. In the Model Explorer, expand the Simulink root node and select **Configuration Preferences**.

You can access the **model-specific configuration sets** by selecting **View - Model Explorer** from your model's menu bar. In the Model Explorer, expand the node of your model and select the configuration set.





You can also access the model-specific configuration set via the **Simulation - Configuration Parameters** menu command of your model.

You can access the **referenced configuration sets** by selecting **Add – Configuration Reference** from your model's menu bar. In the Model Explorer, expand the node of your model and select the **Reference** node.


**Description**

The Model Explorer's Configuration Parameter dialog lets you specify both configuration sets for a particular model's simulation and code generation and referenced configuration sets.

- For details on model-specific configuration sets, refer to *How to Specify RTI-Specific Settings for a Model* .
- For details on referenced configuration sets, refer to *How to Attach a Configuration Reference* .

**Dialog pages**

This dialog contains the following pages:

- **Solver Dialog (Model Configuration Parameters Dialogs)** on page 18 for setting options for the start and stop times, choosing the Solver, and defining the timer task mode.
- **Data Import/Export dialog** for setting options for writing simulation data to a MAT file during **Simulink simulations**. The settings of this page are not relevant to RTI. The ability to capture data in real time for applications built by RTI is provided by dSPACE's ControlDesk (see *Capturing Data*  in the *ControlDesk Experiment Guide*).

- *Optimization Dialog (Model Configuration Parameters Dialogs)* on page 21 for setting options that improve simulation performance and the performance of code generated from your model.
- *Diagnostics Dialog (Model Configuration Parameters Dialogs)* on page 23 for setting options for making specific model conditions generate messages.
- *Hardware Implementation Dialog (Model Configuration Parameters Dialogs)* on page 25 for specifying the characteristics of the hardware to be used to implement the system represented by your model.
- Model Referencing dialog for specifying options for including other models in your model and vice versa, and for building simulation and code generation targets. Since model referencing is not supported by RTI, the settings of this page are not relevant.
- *Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)* on page 27 for specifying the RTI driver programs that control the automatic build process, and for setting different Real-Time Workshop and RTI simulation, build, variable description file, and download options.



---

#### Related topics

##### Basics

- [How to Attach a Configuration Reference](#) 

##### HowTos

- [How to Set Default Values for Build Options](#) 
- [How to Specify RTI-Specific Settings for a Model](#) 

## Solver Dialog (Model Configuration Parameters Dialogs)

---

#### Access

This dialog is contained in the **Model Configuration Parameters** dialog. You can access it by selecting **View - Model Explorer** from your model's menu bar. In the Model Explorer, expand the node of your model and select **Configuration**. Then select **Solver** in the Contents pane.

---

#### Purpose

To specify parameters for the start and stop times, choose a solver, and define the timer task mode for single-processor RTI.

**Description**

In this dialog you can specify the solver-related parameters for real-time simulation and the Simulink simulation.

In this reference you will find only the settings relevant to real-time simulation. For a detailed description of the Solver dialog and its options, refer to the MATLAB documentation by The MathWorks.

**Simulation time frame**

**Start time** Lets you specify a start time for the simulation.

For real-time applications, the start time must always be set to 0.0 seconds.

**Stop time** Lets you specify a stop time for the simulation.

For real-time simulations you should specify `inf` (meaning infinity) as the stop time.

**Solver options frame**

**Type** Specifies the type of solver to be used to solve the currently selected model, either Fixed-step or Variable-step.

**Solver** Lets you specify the solver type for calculating the model.

The higher the order of a solver, the higher its numerical precision and the more execution time it needs. With RTI you can use the following fixed-step solvers:

Solver	Description
discrete	No integration, suitable only for models without continuous states
ode1	Euler's method, suitable for most applications
ode2	Heun's method, also known as the improved Euler formula
ode3	The Bogacki-Shampine formula
ode4	The fourth-order Runge-Kutta formula
ode5	The Dormand-Prince formula
ode14x	Combination of Newton's method and extrapolation from the current value

**Periodic sample time constraint** Lets you specify constraints on the sample times defined by the model. During simulation, Simulink checks to ensure that the model satisfies the constraints. For RTI, this option must be set to **Unconstrained**.

**Fixed step size (fundamental sample time)** Lets you specify the base sample time for the model. If your model includes continuous states, the fixed step size is also used for the integration method.

The fixed step size has to be specified in seconds.

### Accuracy of the simulation step size



The simulation step size during the real-time simulation and the fixed step size selected in the model might differ, depending on the resolution of the timer device that provides the simulation step size. This difference is a discretization error resulting from the approximation of the fixed step size by an integral multiple of the timer resolution. The discretization error is always smaller than the timer resolution and can be minimized by selecting a fixed step size that is an integral multiple of the timer resolution.

Consider the following example:

- Timer resolution = 0.015  $\mu$ s
- Fixed step size = 25  $\mu$ s

In this case the simulation step size is given by:




$$\text{floor}(25 \mu\text{s} / 0.015 \mu\text{s}) * 0.015 \mu\text{s} = 24.99 \mu\text{s}.$$

**Extrapolation order** Lets you specify the extrapolation order used by the ode14x solver to compute a model's states at the next time step from the states at the current time step.

**Number Newton's iterations** Lets you specify the number of Newton's method iterations used by the ode14x solver to compute a model's states at the next time step from the states at the current time step.

**Tasking mode for periodic sample times** Lets you specify the execution mode for calculating your model.

- If "SingleTasking" is selected, the application contains just one timer task. It is suitable for most applications.
- If "MultiTasking" is selected, the application contains the same number of timer tasks as it does sample rates.
- If "Auto" is selected, RTI automatically chooses the execution mode depending on the sample rates of your model: single-rate models are built in the single timer task mode and multirate models are built in the multiple timer task mode.

For details on both execution modes, refer to *Single Timer Task Mode* , *Multiple Timer Task Mode*  and *Comparing the Execution Modes*  in the *RTI and RTI-MP Implementation Guide*.

**Higher priority value indicates higher task priority** If this checkbox is selected, the real-time system targeted by this model assigns a higher priority to tasks with higher priority values. If this checkbox is cleared, the real-time system targeted by this model assigns a higher priority to tasks with lower priority values. For RTI, this checkbox must be cleared.

#### Related topics

HowTos

- *How to Set Default Values for Build Options* 

## Optimization Dialog (Model Configuration Parameters Dialogs)

<b>Access</b>	This dialog is contained in the <b>Model Configuration Parameters</b> dialog. You can access it by selecting <b>View - Model Explorer</b> from your model's menu bar. In the Model Explorer, expand the node of your model and select <b>Configuration</b> . Then select <b>Optimization</b> in the Contents pane.	
<b>Purpose</b>	To set options for signal storage reuse, inline parameters, optimizing block reduction, etc.	
<b>Description</b>	<p>In this dialog you can specify options that improve simulation performance and the performance of code generated from your model.</p> <p>In this reference you will find only the settings relevant to real-time simulation. For a detailed description of the Optimization dialog and its options, refer to the MATLAB documentation by The MathWorks.</p>	
<b>Simulation and code generation frame</b>	<b>Block reduction optimization</b>	This option is not supported by RTI. Make sure that it is not selected.

**Implement logic signals as boolean (vs. double)** Lets you specify whether the boolean or double data type is used for logical signals. This option can be used for compatibility with models from earlier Simulink versions where logical signals are implemented as double.

- If selected, the Simulink blocks (e.g., Logical Operator) use the **Boolean** data type for logical signals (default).
- If cleared, the Simulink blocks in the model use the **double** data type for logical signals.

For reasons of downward compatibility, this setting is also respected by some RTI Bit I/O blocks.

For general information, refer to the Simulink documentation.

**Signal storage reuse** Lets you control the memory allocation for block output variables.

- If selected, Real-Time Workshop does not always use a separate block output variable for each block output but attempts to assign a single buffer to several block outputs. This typically reduces memory consumption. However, reused variables are not available in the variable description file.

You can exclude individual block outputs from this optimization to make them available in the variable description file.

Related files: *Variable Description File (TRC File)* on page 109.

**Inline parameters** Lets you inline the block parameters or generate them as variables.

- If selected, the block parameters are inlined, so their concrete values are used in the generated C code, which makes them non-modifiable. This typically reduces memory consumption and execution time. As a result, no block-specific parameter is available below the **Model Root** node of the variable description file.

To generate individual parameters as tunable variables, see *Configure* below.

- If cleared, the block parameters are generated as variables, so they are modifiable during the real-time simulation (default).

Related files: *Variable Description File (TRC File)* on page 109

**Configure** Lets you make individual parameters tunable if the Inline parameters option is selected. Parameters declared as tunable parameters are available in the variable description file via the Tunable Parameters group.

For details, refer to *Model Parameters Configuration Dialog* on page 37.

Related files: *Variable Description File (TRC File)* on page 109.

## Diagnostics Dialog (Model Configuration Parameters Dialogs)

<b>Access</b>	This dialog is contained in the <b>Model Configuration Parameters</b> dialog. You can access it by selecting <b>View - Model Explorer</b> from your model's menu bar. In the Model Explorer, expand the node of your model and select <b>Configuration</b> . Then select <b>Diagnostics</b> in the Contents pane.
<b>Purpose</b>	To set options for making specific model conditions generate messages.
<b>Description</b>	<p>In this dialog you can specify the diagnostic actions Simulink takes if it detects an error during the compilation or simulation of a model. The Diagnostics dialog contains the following pages:</p> <ul style="list-style-type: none"> <li>■ The <b>Solver</b> page lets you specify the diagnostic actions Simulink takes when it detects a solver-related error.</li> <li>■ The <b>Sample Time</b> page lets you specify the diagnostic actions Simulink takes when it detects an error relating to the sample times in your model.</li> <li>■ <i>Data Validity page</i> on page 24 lets you specify the diagnostic actions Simulink takes when it detects an error that could have an impact on the data integrity of your model.</li> <li>■ The <b>Conversion</b> page lets you specify the diagnostic actions Simulink takes when it detects a data type conversion error during compilation of your model.</li> <li>■ The <b>Connectivity</b> page lets you specify the diagnostic actions Simulink takes when it detects an error relating to block connections.</li> <li>■ The <b>Compatibility</b> page lets you specify the diagnostic actions Simulink takes when it detects an error relating to compatibility between the current Simulink version and your model.</li> </ul>

- The **Model Referencing** page lets you specify the diagnostic actions Simulink takes when it detects an error relating to Model Referencing.

In this reference you will find only the settings relevant to real-time simulation. For a detailed description of the **Diagnostics** dialog and its options, refer to the *Using simulink* by The MathWorks.


#### Data Validity page

**Model Verification block enabling** Lets you globally enable or disable all the Model Verification blocks or use the settings of the individual blocks.

- If “Enable all” is selected, all the **Model Verification** blocks are active.
- If “Disable all” is selected, none of the **Model Verification** blocks is active.
- If “Use local settings” is selected, only the **Model Verification blocks** with the **Enable assertion** setting selected are active (default).

To control the behavior of the Model Verification blocks in the real-time simulation, use RTI’s **Assertion mode** setting. Depending on how you set this, one of the following actions is carried out if a verified signal leaves the desired range, meaning that the assertion occurs:

- “OFF”: No reaction (default).
- “WARN”: Issue a warning message in ControlDesk’s Log Viewer.
- “STOP”: Issue an error message and stop the simulation, i.e., set the simulation state to STOP.

For details on this technique, refer to *How to Use Simulink’s Model Verification Blocks* (  *RTI and RTI-MP Implementation Guide*).



## Hardware Implementation Dialog (Model Configuration Parameters Dialogs)

<b>Access</b>	<p>This dialog is contained in the <b>Model Configuration Parameters</b> dialog. You can access it by selecting <b>View - Model Explorer</b> from your model's menu bar. In the Model Explorer, expand the node of your model and select <b>Configuration</b>. Then select <b>Hardware Implementation</b> in the Contents pane.</p>
<b>Purpose</b>	<p>To set options for the characteristics of the hardware to be used to implement the system represented by your model.</p> <p>In this reference you will find only the settings relevant to real-time simulation. For a detailed description of the Hardware Implementation dialog and its options, refer to the MATLAB documentation by The MathWorks.</p>
<b>Description</b>	<p>The Hardware Implementation dialog lets you specify the characteristics of the hardware to be used to implement the system represented by your model. It contains the following two groups of properties:</p> <ul style="list-style-type: none"> <li>■ <b>Embedded hardware (simulation and code generation)</b> lets you specify the properties of the production-type hardware.</li> <li>■ <b>Emulation hardware (code generation only)</b> lets you specify the properties of the hardware used to test the code generated from your model.</li> </ul> <p>RTI and RTI-MP support code generation for the following configuration types:</p> <ul style="list-style-type: none"> <li>■ The RTI platform represents the production-type hardware, and no emulation hardware is used. You have to perform the following steps for configuration: <ul style="list-style-type: none"> <li>■ Under Embedded hardware, specify the properties applying to your dSPACE board.</li> <li>■ Under Emulation hardware, select <b>None</b>.</li> </ul> </li> <li>■ The behavior of arbitrary production-type hardware is emulated on the RTI platform. You have to perform the following steps for configuration: <ul style="list-style-type: none"> <li>■ Under Embedded hardware, specify arbitrary properties.</li> <li>■ Under Emulation hardware, specify the properties applying to your dSPACE board.</li> </ul> </li> </ul>

The behavior of the hardware specified under Embedded hardware is emulated on the RTI platform.

If you select an RTI platform on the General page of the Real-Time Workshop dialog, the following values are automatically configured on the Hardware Implementation dialog, depending on your dSPACE board:

Property	DS1005, DS1103, DS1104, DS1401	DS1006
Device type	Custom	Custom
Number of bits		
char	8	8
short	16	16
int	32	32
long	32	32
native word size	32	32
Byte ordering	Big Endian	Little Endian
Signed integer division rounds to	Zero	Zero
Shift right on a signed integer as arithmetic shift	True	True
Emulation hardware (code generation only)	None	None

RTI and RTI-MP check if the Hardware Implementation settings are specified correctly.

#### Embedded hardware frame

**Device type** Lets you specify the type of hardware that is used to implement the production version of the system represented by your model.

**Number of bits** Lets you specify the length in bits of C data types supported by the selected device type.

**Byte ordering** Lets you specify the significance of the first byte of a data word of the target hardware.

**Signed integer division rounds to** Lets you specify how an ANSI C-compliant compiler used to compile code for the production hardware rounds the result of dividing one signed integer by another to produce a signed integer quotient.

**Shift right on a signed integer as arithmetic shift** Select this option if the C compiler implements a signed integer right shift as an arithmetic right shift.

<b>Emulation hardware frame</b>	<p><b>None</b> Lets you specify that the hardware used to test the code generated from this model is the same as the production hardware or has the same characteristics.</p> <p>The following options are visible only if the <b>None</b> checkbox is cleared:</p> <p><b>Device type</b> Lets you specify the type of hardware that is used to implement the production version of the system represented by your model.</p> <p><b>Number of bits</b> Lets you specify the length in bits of C data types supported by the selected device type.</p> <p><b>Byte ordering</b> Lets you specify the significance of the first byte of a data word of the target hardware.</p> <p><b>Signed integer division rounds to</b> Lets you specify how an ANSI C-compliant compiler used to compile code for the production hardware rounds the result of dividing one signed integer by another to produce a signed integer quotient.</p> <p><b>Shift right on a signed integer as arithmetic shift</b> Select this option if the C compiler implements a signed integer right shift as an arithmetic right shift.</p>
---------------------------------	--

## Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)

<b>Access</b>	This dialog is contained in the <b>Model Configuration Parameters</b> dialog. You can access it by selecting <b>View - Model Explorer</b> from your model's menu bar. In the Model Explorer, expand the node of your model and select <b>Configuration</b> . Then select <b>Real-Time Workshop</b> in the Contents pane.
<b>Purpose</b>	To specify the RTI driver programs that control the automatic build process; to set different Real-Time Workshop and RTI simulation, build, variable description file, and download options.
<b>Description</b>	On this page you can specify options relating to the code generation process. The following settings are relevant to RTI. For the settings relevant to RTI-MP, refer to <i>Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)</i> on page 75. For a detailed description of the Real-Time Workshop page and its options, refer to Real-Time Workshop User's Guide by The MathWorks.

### Dialog settings

The Real-Time Workshop dialog contains several pages. The following pages contain settings that are relevant to RTI:

- *General page* on page 28
- *Custom Code page* on page 29
- *Interface page* on page 29
- *RTI simulation options page* on page 30
- *RTI general build options page* on page 32
- *RTI load options page* on page 33
- *RTI variable description file options page* on page 34



You can use the `rti_optionget` and `rti_optionset` commands to get or set the options supported by the RTI options pages. Refer to *rti\_optionget* on page 135 and *rti\_optionset* on page 133.


**Generate code only** Lets you generate only C code from your Simulink model.

- If selected, the model code is only generated. The code is neither compiled nor downloaded to the hardware.
- If cleared, the model code is generated, compiled, and downloaded to the dSPACE hardware (default).

**Build/Generate code** Lets you start the build and download procedure.

The button label depends on the state of the **Generate code only** checkbox.

- If the label is "Build", the model code is generated, compiled, linked, and downloaded to the dSPACE hardware.
- If the label is "Generate code", only the model code is generated. The code is neither compiled nor downloaded to the hardware.

For an overview of the build process, refer to *Basics for the Build and Download* .

### General page

**System target file** When a new model is created, RTI automatically selects the correct system target file for your hardware, which you do not need to change. For information on the system target file, refer to *rti[mp]<xxx>.tlc* on page 95.

**Make command** When creating a new model, RTI automatically specifies the correct make command, which you do not need to change.

**Template makefile** When creating a new model, RTI automatically selects the correct template makefile for your hardware, which you do not need to change. For information on the related files, refer to *rti[mp]<xxx>.tmf* on page 95, *<[sub]model>.mk* on page 96, and *<[sub]model>\_usr.mk* on page 96.

## Custom Code page

**Include custom c-code in generated:** Lets you insert code fragments into the generated files.



You can use the settings in the Include custom c-code in generated frame in Real-Time Workshop's Custom Code page in addition to the settings in the RTI User-Code file (USR.C).

**Include list of additional:** Lets you include additional files and paths in the build process.



You can use the settings in the **Include list of additional** frame in Real-Time Workshop's Custom Code page instead of, or in addition to, the settings in the RTI User Makefile (USR.MK).

## Interface page

**Target floating point math environment** Lets you specify which floating-point math library to use during code generation. It is recommended to use the default selection "C89/C90 (ANSI)" to avoid incompatibilities of the generated code with the target compilers used by RTI.

**MAT-file variable name modifier** Lets you select a string which is added to the variable names used when data is logged to MAT files.

**Utility function generation** Lets you select a location for Real-Time Workshop to place utility code. You can use the shared utilities directory by selecting "Shared location" from the Utility function generation list.

**Interface** Lets you specify to have Real-Time Workshop include one of three APIs when it generates code. You can choose between **C-API**, **External mode**, and **ASAP2**.

If you select External mode, the following additional options appear on the Interface page:

- *Transport layer* on page 30
- *MEX-file arguments* on page 30
- *Static memory allocation* on page 30

**Transport layer** Lets you specify a messaging protocol for host/target communications.

**MEX-file arguments** The MEX file arguments are optional and let you specify the target:

- If left blank (default), RTI's external mode interface connects to the working board that was defined in ControlDesk.
- If a board is specified using the syntax: 'BOARD=<board\_name>' and the board is available in the connection defined by the working board (bus/network), the board is used as the target.

MEX file arguments have to be enclosed by single quotes and separated by a comma.

**Static memory allocation** Lets you specify to have Real-Time Workshop generate code for external mode that uses only static memory allocation. If you select this checkbox, the Static memory buffer size edit field appears, which lets you specify the size of the static memory buffer used by external mode.

---

#### RTI simulation options page

**Initial simulation state** Lets you define the **initial simulation state** for a model.

- If set to "RUN", the simulation is started right after the download (default).
- If set to "PAUSE", the simulation is initialized after the download but not started. The output devices are set to the **Initialization** values.
- If set to "STOP", the simulation remains stopped after the download. The output devices are set to the **Termination** values.

The current simulation state of the real-time simulation is available in the variable description file (see *simState* on page 88).

**Execution mode** Lets you specify whether the model should be calculated in real-time or non real-time execution mode.

- If "real-time" is selected, the model is calculated in real time (default).
- If "time-scaled" is selected, the simulation is performed faster or more slowly, meaning that calculating one second in the simulation takes less or more than one real second.  
You have to provide a Time scale factor (see below).
- If "as fast as possible" is selected, the simulation is performed as fast as possible without a specific time scale factor.

To execute your model in the “time-scaled” or “as fast as possible” execution modes, you have to build it in the single timer task mode. You can still use hardware and software interrupts.



If you use an execution mode other than real-time, you must turn off the time-stamping feature in ControlDesk. Otherwise, the x-axes of the plots in ControlDesk do not match the simulation time. This is because time stamps are derived from a real-time clock, which always delivers time stamps in real time regardless of whether the application is executed in real time or not.


**Time scale factor** Lets you specify a time scale factor for the “time-scaled” execution mode. (Enabled only if the “time-scaled” execution mode is selected, see above.)

For example, if you specify the “time-scaled” execution mode and time scale factor 0.2, the simulation runs 5 times faster than in the “real-time” execution mode. A time scale factor of 1 corresponds to a real-time simulation.

**Assertion mode** RTI supports Simulink’s Model Verification blocks for real-time simulation.

To control the behavior of the Model Verification blocks in the real-time simulation, use RTI’s **Assertion mode** setting. Depending on how you set this, one of the following actions is carried out if a verified signal leaves the desired range, meaning that the assertion occurs:

- “OFF”: No reaction (default).
- “WARN”: Issue a warning message in ControlDesk’s Log Viewer.
- “STOP”: Issue an error message and stop the simulation, i.e., set the simulation state to STOP.

For details on this technique, refer to *How to Use Simulink’s Model Verification Blocks* . See also *Model Verification block enabling* on page 74.

**Enable Test Automation Stimulus Engine** Lets you enable test automation.

- If selected, you can use ControlDesk’s stimulus signal generation feature with the generated real-time application.
- If cleared, you cannot use ControlDesk’s stimulus signal generation feature with the generated real-time application (default).

**Enable real-time testing** Lets you enable or disable real-time testing for the generated real-time application.



This option is only available for RTI1005/1006.

**Task Configuration** Lets you assign priorities to the tasks of a model and specify their overrun behavior.

This opens the RTI Task Configuration dialog, which lets you configure the tasks of a model. Refer to *RTI Task Configuration Dialog* on page 38.

#### RTI general build options page

**Compiler options** Lets you specify the compiler options.

For example, if you want the compiler to produce debugging information. Specify the following compiler option: `-g`

**Compiler optimizations** Lets you specify the compiler optimization level.

- If “Default” is selected, the model is compiled with the platform- and compiler-specific default optimization, which is full inline expansion and the highest recommended level of optimization.
- If “User defined” is selected, you can define the optimization via the User-defined optimization edit field.
- If “None” is selected, the compilation is performed without any optimization.

**Effective Optimizations** Shows the optimization settings currently used for the selected compiler optimization level.

**User-defined optimization** Lets you specify user-defined compiler options. (Available only if compiler optimizations is set to “User-defined”.)

- If empty, the model code is compiled with the default optimization setting of the compiler.



- You should not enter optimization levels higher than the default. For information about the permissible compiler options, refer to the following manuals:

Hardware	Manual
PowerPC-based (DS1005, DS1103, DS1104, MicroAutoBox)	<i>C/C++ Compiler User's Guide by Microtec</i>
x86-based (DS1006)	<a href="http://gcc.gnu.org/onlinedocs/">http://gcc.gnu.org/onlinedocs/</a>

**Enable data set storage in application** Lets you enable the storing of new data sets in an application image file.

- If the checkbox is selected, you can use ControlDesk or CalDesk to create a new application image file containing a new data set without rebuilding the application.
- If the checkbox is cleared, you cannot use ControlDesk or CalDesk to create a new application image file containing a new data set without rebuilding the application (default).



This option is only available for RTI1005 and RTI1401.

## RTI load options page

**Load application after build** Loads the real-time application to the dSPACE hardware after it is built.

- If selected, the real-time application is automatically downloaded to the dSPACE hardware after it is built.
- If cleared, the real-time application is not downloaded automatically.

**Load to flash memory** Lets you download a real-time application to the flash memory.

- If selected, the real-time application is downloaded to the flash memory (nonvolatile memory).
- If cleared, the real-time application is downloaded to the RAM (default).

**Platform selection** Lets you select a destination for downloading the real-time application.

- If "Auto" is selected, RTI downloads the real-time application to the Working Board you specified in ControlDesk, provided that the board matches the current real-time application.

- If “User-defined (bus)” is selected, RTI downloads the real-time application to the processor board specified in the Processor board name option. The board must be connected to the host PC via the bus and registered with ControlDesk’s Platform Manager.
- If “User-defined (network)” is selected, RTI downloads the real-time application to the processor board specified in the Processor board name option using the specified Network client option. The board must be connected to the host PC via the specified network client and registered with ControlDesk’s Platform Manager.

**Processor board name** Lets you specify the processor board to download the real-time application to (available only if the platform selection is set to “User-defined (bus)” or “User-defined (network)”).

**Network client** Lets you specify the network client used for downloading the real-time application (available only if the platform selection is set to “User-defined (network)”).

Not available for all platforms.

**Slave processor object** Lets you download an additional application to the slave processor.

You can specify the slave object file with a relative or absolute path. Do not specify the file name extension for the object file.

For example, to specify the 4df slave object file with a relative path, enter 4df. To specify a slave object file with an absolute path, you can use environment variables. Put the environment variable into \$(), for example, \$(DSPACE\_ROOT)\slave\4df.


#### RTI variable description file options page

**Include mask and workspace parameters** Lets you make mask and workspace parameters available in the **variable description file**.

- If selected, mask and workspace parameters are available in the variable description file. The dependent parameters are set to read-only.
- If cleared, mask and workspace parameters are not available in the variable description file. The dependent parameters are writable (default).



Using the TRC Exclusion Block in conjunction with the **Include mask and workspace parameters** option is not supported.

You can change any number of dependent parameters by means of just one mask or workspace parameter. For details on this technique, see *Using Workspace and Mask Parameters* .



If you work with mask and workspace parameters, you have to consider some limitations. See *Using Workspace and Mask Parameters*.

Further restrictions:

- MATLAB must be installed on the host PC running the ControlDesk experiment.
- Multidimensional parameters are not supported. For example, if you have a 3x4x2 Lookup Table (n-D) block, this comes out as one `LookupTableData` entry of the dimension 24x1 in the variable description file.

Related files: *Variable Description File (TRC File)* on page 109

**Include signal labels** Lets you make the signal labels of the model available in the variable description file.

- If selected, the signal labels of the model are generated into the variable description file (default).
- If cleared, the signal labels of the model are not generated into the variable description file. For large models especially, this might significantly reduce the time needed for the code generation process.

Related files: *Variable Description File (TRC File)* on page 109

**Include virtual blocks** Lets you make the block outputs of virtual blocks (Mux, Demux, From, etc.) available in the variable description file.

- If selected, the virtual blocks are available in the variable description file (default).
- If cleared, the virtual blocks are not available in the variable description file. This might significantly reduce the time needed for the code generation process.

If you just want to have the outputs of a few virtual blocks available in the variable description file, you can also insert a unity Gain block (**Gain** value set to 1). Its output is available in the variable description file, even if you turn off the generation of virtual blocks.

Related files: *Variable Description File (TRC File)* on page 109

**Include states** Lets you make the states of blocks available in the variable description file.

- If selected, the block states are generated into the variable description file.
- If cleared, the block states are not generated into the variable description file (default).


Related files: *Variable Description File (TRC File)* on page 109

**Include derivatives** Lets you make the derivatives of blocks available in the variable description file.

- If selected, the derivatives of blocks are generated into the variable description file.
- If cleared, the derivatives of blocks are not generated into the variable description file (default).

Related files: *Variable Description File (TRC File)* on page 109

**Apply subsystem read/write permissions** Lets you apply Simulink's ReadOnly and NoReadNoWrite permissions for subsystems to the variable description file.

- If selected, the parameters of subsystems with the ReadOnly permission are set to `read-only` in the variable description file. The content of NoReadOrWrite subsystems is hidden in ControlDesk, only the subsystem itself is visible, including its mask parameters (if any).
- If cleared, the parameters of all subsystems are visible and writable in the variable description file, with the following exceptions:
  - A subsystem can be excluded via the **TRC Exclusion** block. Refer to *Excluding Subsystems from the TRC File* .
  - The parameters of masked subsystems are read-only if the **Include mask and workspace parameters** option is selected.



You can configure ControlDesk to show the contents of NoReadNoWrite subsystems.

Related files: *Variable Description File (TRC File)* on page 109

**Include initial parameter values** To make the initial parameter values available in the variable description file.

- If selected, the initial parameter values of the model are generated into the variable description file (this does not apply to the Stateflow parameters of the State Machine Data group). This lets you carry out offline calibration for your model with CalDesk.

- If cleared, the initial parameter values of the model are not generated into the variable description file. For large models especially, this can significantly reduce the time needed for the code generation process (default).

Variable description files generated for ControlDesk-controlled Simulink simulations do not contain the initial parameter values, regardless of whether this option is selected or not.

## Model Parameters Configuration Dialog

<b>Access</b>	You can access this dialog via the Configure button, which is located in the Optimization dialog of the Configuration Parameters dialog in the Model Explorer. Refer to <i>Optimization Dialog (Model Configuration Parameters Dialogs)</i> on page 21.
<b>Purpose</b>	To declare specific parameters as tunable parameters.
<b>Description</b>	You can make individual parameters tunable if the Inline parameters option is selected. The fewer parameters you declare tunable, the more memory and execution time can be saved.
<b>Dialog settings</b>	This section gives only a brief summary of the dialog settings. For a detailed description of the <b>Tunable Parameters</b> dialog and its options, refer to <i>Using Simulink</i> by The MathWorks.
<b>Global (tunable) parameters frame</b>	<p><b>New</b> Adds a new tunable parameter.</p> <p><b>Remove</b> Removes a tunable parameter.</p> <p><b>Name</b> Enter the parameter name here.</p> <p><b>Storage class</b> Select the storage class for the parameter.</p> <p><b>Storage type qualifier</b> Enter the storage type qualifier here.</p>
<b>Source List frame</b>	From the drop-down list, select whether you want to have all "MATLAB workspace" variables or only the "Referenced workspace variables" displayed in the source list.

The source list displays all the workspace variables that correspond to the selection of the drop-down list. The variables that are made global (tunable) parameters are displayed in bold italics.

**Refresh list** Updates the list from the workspace/model.

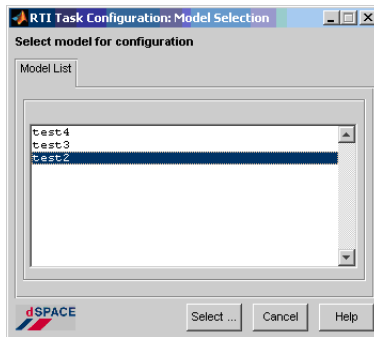
**Add to table >>** Adds the highlighted variable(s) to the Global (tunable) parameters list.

## RTI Task Configuration Dialog

### Access

You can access this dialog via the Task Configuration button (located on the Real-Time Workshop page of the **Simulation Parameters** dialog with “RTI simulation options” as the selected category). Refer to *RTI simulation options page* on page 30.

When working with configuration references, the Task Configuration button of a configuration set in the MATLAB base workspace that is referenced by at least two open models opens a **Model Selection** dialog.



The dialog lists all open models that have an activated configuration reference to the current configuration set in the MATLAB base workspace. Click **Select** to select the model you want to open the RTI Task Configuration dialog for.

### Purpose

To assign priorities to the tasks of a model and configure the overrun strategy.

### Description

You can assign a priority to each task of your model and configure its overrun strategy. Both are relevant only to real-time simulation.

---

**Dialog settings**

**Tasks with configurable priority** Displays the tasks of your model.

- Priorities are ordered top-down.
- Priority numbers can range from 1 (highest priority) to 128 (lowest priority).
- The background task is not shown because it always has the lowest priority.
- Interrupt-driven tasks are named after the corresponding interrupt blocks.
- Timer tasks are named after their sample times.

The number of timer tasks depends on the execution mode. For details, refer to *Tasking mode for periodic sample times* on page 20.

---

**Up** Increases the priority of the highlighted task.

**As Previous** Assigns the priority of the previous task to the currently selected task.

**Down** Decreases the priority of the highlighted task.

**Task name** Displays the name of the currently selected task.

---

**Interrupt source** Displays the name of the interrupt source of the currently selected task. This can be either a hardware or a software interrupt or a timer interrupt.

**Interrupt block** Displays the name and path of the hardware or software interrupt block if an interrupt block-driven task is selected.

**Priority** Displays the priority of the currently selected task.

**Stop simulation** The simulation is stopped whenever a task overrun occurs.

**Queue task before simulation stop** When a task overrun occurs, the task instance is queued for later execution.

**Max. queued task instances** Maximum number of queued task instances. If this limit is reached, the simulation is stopped.

**Ignore overrun** When a task overrun occurs, the task concerned is not scheduled. As a result, task instances are discarded in overrun situations.



If one of the overrun strategies **Queue task before simulation stop** or **Ignore overrun** is selected, a task overrun might lead to unexpected simulation results since tasks are not executed in real-time.

#### Related topics

##### HowTos

- [How to Change the Task Priorities](#)
- [How to Handle Overrun Situations](#)

##### References

- *priority* on page 86

## External Mode Control Panel

#### Access

You can access this dialog via the **Tools – External Mode Control Panel** menu command of the Simulink model.

#### Purpose

To specify options for the external simulation.

#### Description

For details on the external simulation mode, refer to *External Simulation* in the *RTI and RTI-MP Implementation Guide*.

#### Dialog settings

**Connect/Disconnect** Toggles the external simulation:

- **Connect** switches the Simulink simulation mode from *normal* to *external* and the target interface for external mode connects to the real-time application. After the connection is established, the entire set of the current parameter values of the Simulink model is downloaded. This ensures that the Simulink model and the corresponding real-time application use the same set of parameters.



If parameters of the application were already altered with ControlDesk, these parameter changes are overwritten with the current parameters values of the Simulink model when you start the external simulation.



- **Disconnect** leaves the simulation set to external, but parameter changes in the current Simulink model are no longer passed to the hardware.



Starting and stopping the external simulation does not affect the simulation state of the real-time simulation. To start or stop the real-time simulation, you can change the `simState` variable via ControlDesk, for example.

**Start/Stop Real-Time Code** Not supported.

**Arm Trigger** Not supported. Leave it cleared.

**Enable data uploading** This checkbox works as an Arm trigger button for floating scopes. For details, refer to the MATLAB documentation.

**Duration** Lets you specify the duration for floating scopes.

**Batch download** Lets you select the download mode.

- If selected, parameter changes in the Simulink model are stored and not passed on to the hardware until the **Download** button below the checkbox is clicked.
- If cleared, parameter changes in the current Simulink model are immediately passed on to the hardware.

**Download** Parameter changes that have been stored are passed on to the hardware (enabled only if Batch download is selected).

**Signal & triggering** Not supported.

**Data archiving** Not supported.

## Related topics

Basics

- [External Simulation](#)

HowTos

- [How to Run an External Simulation](#)

References

- [Real-Time Workshop Dialog \(Model Configuration Parameters Dialogs\)](#) on page 27

## Signal Properties Dialog

<b>Access</b>	You can access this dialog via the <b>Edit – Signal Properties</b> menu command of the <b>Simulink</b> model (available only if a signal is selected in the model).	
<b>Purpose</b>	To specify the properties of the selected signal.	
<b>Description</b>	<p>You can make a signal displayable even if certain code optimization options are selected.</p> <p>In this reference you will find only the settings relevant to real-time simulation. For a detailed description of the <b>Signal Properties</b> dialog and its options, refer to <i>Using Simulink</i> by The MathWorks.</p> <p>The Simulink Signal Properties dialog contains the following pages:</p> <ul style="list-style-type: none"><li>■ <i>Logging and accessibility page</i> on page 42</li><li>■ <i>Real-Time Workshop page</i> on page 42</li><li>■ <b>Documentation</b> page (not relevant to RTI)</li></ul> <p><b>Signal name</b>    Specify the name of the signal if desired.</p>	
<b>Logging and accessibility page</b>	<b>Test Point</b>	Select this checkbox if you want to make the signal global.
<b>Real-Time Workshop page</b>	<b>RTW storage class</b>	Select the storage class for the signal if desired.

# RTI-MP Dialog Reference

The various dialogs of Simulink and RTI-MP provide many options to customize the behavior of Simulink and code generation for the dSPACE real-time platforms.

The following dialogs are relevant when you use RTI-MP:

Purpose	Refer to
Interprocessor communication	
To get information on the communication channels of a model	<i>Communication Channels Display</i> on page 44
To assign an IPC block to a different communication channel	<i>Change Connection Dialog</i> on page 44
Model configuration, multiprocessor setup and simulation parameters	
To customize and start automatic program building for multiprocessor systems.	<i>Multiprocessor Setup Dialog</i> on page 45
To specify priorities for the tasks of a model.	<i>RTI Task Configuration Dialog (Multiprocessor Setup Dialog)</i> on page 56
To specify simulation parameters for multiprocessor models.	<i>CPU Options Dialog</i> on page 58
To rename a CPU.	<i>Rename CPU Dialog</i> on page 64
To specify the network topology for a model.	<i>Multiprocessor Topology Setup Dialog</i> on page 65
To specify configuration parameters for multiprocessor models.	<i>Model Configuration Parameters Dialogs</i> on page 67
To make individual parameters tunable if the Inline parameters option is selected.	<i>Model Parameters Configuration Dialog</i> on page 76

Purpose	Refer to
External simulation	
To control an external simulation.	<i>External Mode Control Panel</i> on page 77
Signal properties	
To make a signal displayable even if certain code optimization options are selected.	<i>Signal Properties Dialog</i> on page 78
Related Topics	
The dialogs described here relate to various files.	<i>RTI and RTI-MP File Reference</i> on page 93
For general information on Simulink's dialogs, refer to the corresponding documentation provided by The MathWorks.	

## Communication Channels Display

<b>Access</b>	You can access this dialog via the Show all IPC channels of the model button, which is located on the Advanced page of the <b>Multiprocessor Setup</b> dialog.
<b>Purpose</b>	To display all communication connections and their parameters.
<b>Description</b>	All communication connections and their parameters can be changed by double-clicking the corresponding IPC blocks in the Simulink model.
<b>Related topics</b>	<p>References</p> <ul style="list-style-type: none"> <li>• <i>Advanced Page (Multiprocessor Setup Dialog)</i> on page 54</li> </ul>

## Change Connection Dialog

<b>Access</b>	You can access this dialog via the Change connection button, which is located in the <b>Communication Channel Setup</b> dialog (see <i>Interprocessor Communication (IPC) Block</i> on page 174).
<b>Purpose</b>	To change the CPU names and channel number of the specified IPC block.

**Dialog settings**

**Connection: Source CPU** Lets you specify the name of the source CPU (origin of the **IPC block**). CPU names are restricted to eight alphanumeric characters.

To add another CPU to the multiprocessor model, simply enter a new CPU name. The parameters of the new CPU, such as application name and solver, can be entered the next time the **Multiprocessor Setup** dialog is opened.

**Connection: Destination CPU** Lets you specify the name of the target CPU. CPU names are restricted to eight alphanumeric characters.

To add another CPU to the multiprocessor model, simply enter a new CPU name. The parameters of the new CPU, such as application name and solver, can be entered the next time the Multiprocessor Setup dialog is opened.

**Connection: Channel** Lets you specify the channel number.

You can select any non-negative integer number as the channel number.

**Related topics**

## References

- *Interprocessor Communication (IPC) Block* on page 174

## Multiprocessor Setup Dialog


**Access**

You can access this dialog by double-clicking the **Multiprocessor Setup** block, which is located on the root level of **RTI-MP** models.

**Dialog pages**

This dialog contains the following pages:

- *Main Page (Multiprocessor Setup Dialog)* on page 46 for setting the global parameters of the multiprocessor Simulink model.
- *CPU's Page (Multiprocessor Setup Dialog)* on page 49 for setting the CPU-specific parameters of the multiprocessor Simulink model.
- *Advanced Page (Multiprocessor Setup Dialog)* on page 54 for changing the target multiprocessor system and setting advanced global options.
- *Documentation Page (Multiprocessor Setup Dialog)* on page 56 for getting further information about your model and adding comments.




<b>Description</b>	Automatic program building for multiprocessor systems can be customized and started from the <b>Multiprocessor Setup</b> dialog. The settings on this dialog are necessary to convert a Simulink model into a real-time application.
<b>Related topics</b>	<p>HowTos</p> <ul style="list-style-type: none"> <li>• <i>How to Specify Options for the Build Process</i> </li> </ul> <p>References</p> <ul style="list-style-type: none"> <li>• <i>Model Configuration Parameters Dialogs</i> on page 67</li> <li>• <i>Multiprocessor Setup Block</i> on page 180</li> </ul>

## Main Page (Multiprocessor Setup Dialog)

<b>Access</b>	This page is contained in the <b>Multiprocessor Setup</b> dialog. You can access it by double-clicking the <b>Multiprocessor Setup</b> block, which is located on the root level of <b>RTI-MP</b> models.
<b>Purpose</b>	To configure the global options for the real-time code generation of a Simulink model for a multiprocessor system, display the main settings of the application for each CPU, and start the build and download process.
<b>Simulation options frame</b>	<p><b>Initial simulation state</b> Lets you define the <b>initial simulation state</b> for a model.</p> <ul style="list-style-type: none"> <li>■ If set to “RUN”, the simulation is started right after the download (default).</li> <li>■ If set to “STOP”, the simulation remains stopped after the download. The output devices are set to the <b>Termination</b> values.</li> </ul> <p>The current simulation state of the real-time simulation is available in the variable description file (see <i>simState</i> on page 88).</p> <p><b>Stop time</b> Lets you specify a stop time for the simulation.</p> <p>For real-time simulations you should specify <code>inf</code> (meaning infinity) as the stop time.</p>

**Scheduler mode** Lets you specify the execution mode for calculating your model.

- If “single timer task” is selected, the application of each CPU contains just one timer task. It is suitable for most applications (default).
- If “multiple timer task” is selected, the application of each CPU contains the same number of timer tasks as it does sample times.

For details on both execution modes, refer to *Single Timer Task Mode* , *Multiple Timer Task Mode*  and *Comparing the Execution Modes*  in the *RTI and RTI-MP Implementation Guide*.

**Basic step size** Lets you specify the basic step size for a multiprocessor system. You must specify the basic step size as the greatest common divisor for the step sizes of all CPUs.

The step size of a particular CPU equals the product of the basic step size and the step size multiple of that CPU.

The synchronous timing of multiprocessor systems is possible only if the step sizes of all the CPUs have a common divisor: the **basic step size**. You can define an individual step size for each CPU (see page *Step size* on page 48). The product of the **basic step size** and a CPU's **step size** multiple defines the actual step size for the CPU, which is also available in the corresponding variable description files (see *modelStepSize* on page 84). The unit of the basic step size is seconds.

**Execution mode** Lets you specify whether the model must be calculated in real-time or non real-time execution mode.

- If “real-time” is selected, the model is calculated in real time (default).
- If “time-scaled” is selected, the simulation is performed faster or more slowly according to the Time scale factor specified (see below). This means that calculating one second in the simulation takes less or more than one real second.

RTI-MP does not provide an “as fast as possible” setting because multiprocessor applications require well-defined synchronization among all CPUs.

To execute your model in the “time-scaled” execution modes, you have to build it in the single timer task mode. You can still use hardware and software interrupts.



If you use an execution mode other than real-time, you must turn off the time-stamping feature in ControlDesk. Otherwise, the x-axes of the plots in ControlDesk do not match the simulation time. This is because time stamps are derived from a real-time clock, which always delivers time stamps in real time regardless of whether the application is executed in real time or not.

**Time scale factor** Lets you specify a time scale factor for the “time-scaled” execution mode. (Enabled only if the “time-scaled” execution mode is selected, see above.)

For example, if you specify the “time-scaled” execution mode and time scale factor 0.2, the simulation runs 5 times faster than in the “real-time” execution mode. A time scale factor of 1 corresponds to a real-time simulation.

#### Application setup frame

**CPU** Displays the names of the CPUs that are defined in the model. The master CPU is marked with an **(M)**.

**Application** Displays the names of the applications assigned to the CPUs of the model. You can change the application name of a CPU via the Application name setting on the *CPU's Page (Multiprocessor Setup Dialog)* on page 49 (see page *Application name* on page 50).

**Board type** Displays the board types for the CPUs of the model. To change the current settings, see *Board type* on page 50.


**Solver** Displays the selected solvers for the CPUs of the model. To change the current settings, see *Solver* on page 48.

**Step size** Displays the real-time step sizes (*step size multiple* × *basic step size*) for the CPUs of the model. To change the current settings, see *Step size* on page 51 and *Basic step size* on page 47.

**Build/Build OK** Lets you start the build procedure for the selected CPU.



- If the label is “Build”, the application for the selected CPU still needs to be built.
- If the label is “Build OK”, the application for the selected CPU matches the **RTI-MP** configuration.




For an overview of the build process, refer to *Basics for the Build and Download*  in the *RTI and RTI-MP Implementation Guide*.

**Build All** Lets you start the build procedure for all CPUs. This is the same as building the applications for all the CPUs successively.







**Download** Lets you generate the system description file for the model and download the application to the multiprocessor system.

If you provide a user system description file with user-specific statements, **RTI-MP** automatically merges it into the system description file. For details, refer to *How to Download DS230x Applications*  in the *RTI and RTI-MP Implementation Guide* and *Creating System Description Files*  in the *ControlDesk Experiment Guide*.

For an overview of the build process, refer to *Basics for the Build and Download*  in the *RTI and RTI-MP Implementation Guide*.

Related topics

Basics

- *Basics for the Build and Download* 
- *Comparing the Execution Modes* 
- *Creating System Description Files* 
- *Multiple Timer Task Mode* 
- *Simulation Control (RUN/STOP Mechanism)* 
- *Single Timer Task Mode* 

HowTos

- *How to Download DS230x Applications* 

## CPU's Page (Multiprocessor Setup Dialog)

Access

This page is contained in the **Multiprocessor Setup** dialog. You can access it by double-clicking the **Multiprocessor Setup** block, which is located on the root level of **RTI-MP** models.

Purpose

To configure the real-time code generation options of the specified CPU in a multiprocessor Simulink model and define the application names, the solver, the real-time step size and the build options for this CPU. You will find a separate CPU's page for any CPU in your multiprocessor system.

**Basic options frame**

**Application name** Lets you specify the name of the application (= *submodel*) running on the selected CPU.



The name specified in this field is used for the object file and other files that are generated by RTI-MP and Real-Time Workshop. It must therefore be a valid file name, which means that only alphanumeric characters are allowed.

**Board type** Lets you select the board type for all CPUs of the model. If you change the board type for one CPU, all the other CPUs are updated to the same board type automatically. The following board types are available for multiprocessor systems:

Board Type	Processor
DS1005 <sup>1)</sup>	PowerPC 750
DS1006 <sup>1)</sup>	AMD Opteron™ Processor Model 2xx

<sup>1)</sup> DS1005 PPC Boards and DS1006 Processor Boards cannot be mixed within a multiprocessor system using RTI-MP.

RTI-MP automatically chooses the matching *system target file* controlling the generation of the application code, and the template makefile controlling the compilation and linkage process for all source modules of an application (see *rti[mp]<xxx>.tlc* on page 95).

**Solver** Lets you specify the solver type for calculating the submodel.

The higher the order of a solver, the higher its numerical precision and the more execution time it needs. With **RTI-MP** you can use the following solvers:

Solver	Description
Decentral integration solvers	
discrete	No integration, suitable only for models without continuous states
ode1	Euler's method, suitable for most applications
ode2 <sup>1)</sup>	Heun's method, also known as the improved Euler formula
ode3 <sup>1)</sup>	The Bogacki-Shampine formula
ode4 <sup>1)</sup>	The fourth-order Runge-Kutta formula
ode5 <sup>1)</sup>	The Dormand-Prince formula
ode14x <sup>1)</sup>	Combination of Newton's method and extrapolation from the current value
Central integration solvers	
ode2-mp <sup>2)</sup>	Heun's method, also known as the improved Euler formula
ode3-mp <sup>2)</sup>	The Bogacki-Shampine formula
ode4-mp <sup>2)</sup>	The fourth-order Runge-Kutta formula
ode5-mp <sup>2)</sup>	The Dormand-Prince formula

<sup>1)</sup> In multiprocessor models, the CPUs communicate once per simulation step. If a model has strong dynamic coupling between the model parts calculated on different CPUs, this might result in significant errors and even instability. In this case, you should consider a central integration solver.

<sup>2)</sup> The CPUs also communicate during the intermediate steps of the integration algorithm. This results in higher stability, especially for models with strong dynamic coupling between the model parts calculated on different CPUs. You must use the same integration algorithm and step size for all CPUs.

**Extrapolation order** Lets you specify the extrapolation order used by the ode14x solver to compute a model's states at the next time step from the states at the current time step.

**Number Newton's iterations** Lets you specify the number of Newton's method iterations used by the ode14x solver to compute a model's states at the next time step from the states at the current time step.

**Step size** Lets you specify the step size multiples for the step sizes of the individual CPUs.

The actual step size of the selected CPU is the product of the step size multiple and the Basic step size (see page *Basic step size* on page 47).

The step size multiple of a CPU must be an integer multiple of the step size multiples of all other CPUs with a smaller step size multiple. For example, if your system contains two CPUs and the step size multiple of CPU1 is 2, you can specify 4 as the step size multiple for CPU2.

#### Advanced options frame

**Profile synchronized swinging buffer communication** Lets you profile all the interprocessor communication channels that are operated with the synchronized swinging buffer protocol.

- If selected, the `rtimpdiag` diagnostic tool provides additional profiling information for all the interprocessor communication input channels of the current CPU that are operated with the synchronized swinging buffer protocol. This includes the number of input calls, the current waiting time, the maximum waiting time, and whether an overrun is caused while waiting.


Profiling the interprocessor communication causes a negligibly small time overhead for each IPC input channel that is operated with the synchronized swinging buffer protocol.

- If cleared, the time overhead is eliminated. As a result, `rtimpdiag` does not provide the profiling information (default).

**Treat as optional CPU** Lets you make the selected CPU optional.

- If selected, the resulting real-time application can be executed even if the current CPU is unavailable, for example, switched off or removed from the multiprocessor system.
- If cleared, the current CPU must be available when you want to run the resulting real-time application (default).

Making certain slave CPUs optional lets you reuse the same real-time application for different hardware configurations. As a result, you do not need to rebuild your model if you do not have all the slave CPUs available.

There are some points to note when using this technique. For details, see *Working with Subsets of a Multiprocessor Topology*  in the *DS1005 Features* document.

**Use Real-TimeWorkshop Custom Code settings** Lets you specify to have the Custom Code page settings of the main model transferred to the relevant submodel.

**Gigalink background CRC** Lets you specify the behavior of Gigalink cyclic redundancy check in the background. You can choose between the following settings:

- Off
- Warn at checksum mismatch
- Error at checksum mismatch



To avoid a timeout during initialization, you can use the following compiler option:

`-DRTIMP_GL_INIT_NUM_RETRIES=n`

### Submodel configuration frame

**Rename CPU** Lets you rename a CPU in the **RTI-MP** model.

This opens the **Rename CPU** dialog, which lets you enter a name for the selected CPU. Refer to *Rename CPU Dialog* on page 64.

**Show submodel** Lets you extract all the blocks that are executed on the specified CPU and place them in a temporary Simulink model.

**Task Configuration** Lets you assign priorities to the tasks of a submodel and specify their overrun behavior.

This opens the **RTI Task Configuration** dialog, which lets you configure the tasks of a submodel. Refer to *RTI Task Configuration Dialog (Multiprocessor Setup Dialog)* on page 56.

### Build process frame

**Build Options** Lets you configure compiler and other build options for the selected CPU.


This opens the **CPU Options** dialog on the **Build Options** page, which lets you specify the settings. Refer to *Build Options Page (CPU Options Dialog)* on page 58.

**Variable description file options** Lets you configure the **variable description file options** for the selected CPU.

This opens the CPU Options dialog on the Variable Description File Options page, which lets you specify the settings. Refer to *Variable Description File Options Page (CPU Options Dialog)* on page 61.

**Build/Build OK** Lets you start the build procedure for the selected CPU.

- If the label is "Build", the application for the selected CPU still needs to be built.
- If the label is "Build OK", the application for the selected CPU matches the **RTI-MP** configuration.

For an overview of the build process, refer to *Basics for the Build and Download*  in the *RTI and RTI-MP Implementation Guide*.

#### Related topics

Basics

- *Handling Exceptions* 

References

- *Files Controlling the Build and Download Process* on page 95

## Advanced Page (Multiprocessor Setup Dialog)

#### Access

This page is contained in the **Multiprocessor Setup** dialog. You can access it by double-clicking the **Multiprocessor Setup** block, which is located on the root level of **RTI-MP** models.

#### Purpose

To define global **RTI-MP** settings, such as the multiprocessor target and the network client.

#### Communication and topology configuration frame

**Multiprocessor network topology** Lets you specify the topology of a multiprocessor system.

This opens the **Multiprocessor Topology Setup** dialog, which lets you configure the network topology. Refer to *Multiprocessor Topology Setup Dialog* on page 65.

**Show all IPC channels of the model** Lets you view all communication connections and their parameters.

This opens the Communication Channels display. Refer to *Communication Channels Display* on page 44.

**Check multiprocessor model** Lets you analyze the multiprocessor model and get a report of possible problems.

#### Load options frame

**Platform selection** Lets you select the platform to download the real-time application to.

- If “Auto” is selected, RTI-MP downloads the real-time application to the Working Board you specified in ControlDesk, provided that this matches the current real-time application.

- If “User-defined (bus)” is selected, RTI-MP downloads the real-time application to the multiprocessor system specified in the Multiprocessor system name option. The multiprocessor system must be connected to the host PC via the bus and registered with ControlDesk’s Platform Manager.
- If “User-defined (network)” is selected, RTI-MP downloads the real-time application to the multiprocessor system specified in the Multiprocessor system name option using the specified Network client option. The multiprocessor system must be connected to the host PC via the specified network client and registered with ControlDesk’s Platform Manager.

**Multiprocessor system name** Lets you select a destination for downloading the real-time application (available only if the Platform selection is set to “User-defined (bus)” or “User-defined (network)”).

**Network client** Lets you specify the network client used for downloading the real-time application (available only if the platform selection is set to “User-defined (network)”).

**Load to flash memory** Lets you download a real-time application to the flash memory.

- If selected, the real-time application is downloaded to the flash memory (nonvolatile memory).
- If cleared, the real-time application is downloaded to the RAM (default).



Downloading a multiprocessor application to the flash memories of the processor boards can take rather a long time, which usually results in a timeout. You can prevent this via the `-DRTIMP_GL_NO_TIMEOUT` option (see *Compiler options* on page 58).

## Related topics

### Basics

- [Limitations for Mask and Workspace Parameters](#)
- [Using Workspace and Mask Parameters](#)

### References

- [Build Options Page \(CPU Options Dialog\)](#) on page 58

## Documentation Page (Multiprocessor Setup Dialog)

<b>Access</b>	This page is contained in the <b>Multiprocessor Setup</b> dialog. You can access it by double-clicking the <b>Multiprocessor Setup</b> block, which is located on the root level of RTI-MP models.
<b>Purpose</b>	To show additional information about the current model.
<b>Dialog settings</b>	<p><b>Type in your comments here</b> Lets you specify comments for the current RTI-MP model.</p> <p><b>Hyperlink to URL</b> Lets you specify a URL for the current RTI-MP model.</p> <p><b>Open URL</b> Lets you open the URL specified as <b>Hyperlink to URL</b>.</p> <p><b>Print Report</b> Lets you generate and open a report for the current RTI-MP model containing all the parameters of the <b>RTI-MP</b> dialogs.</p>

## RTI Task Configuration Dialog (Multiprocessor Setup Dialog)

<b>Access</b>	You can access this dialog via the Task Configuration button (located on the CPU's page of the <b>Multiprocessor Setup</b> dialog). Refer to <i>CPU's Page (Multiprocessor Setup Dialog)</i> on page 49.
<b>Purpose</b>	To assign priorities to the tasks of a model and configure the overrun strategy.
<b>Description</b>	You can assign a priority to each task of your model and configure its overrun strategy. Both are relevant only to real-time simulation.
<b>Dialog settings</b>	<p><b>Tasks with configurable priority</b> Displays the tasks of your model.</p> <ul style="list-style-type: none"> <li>■ Priorities are ordered top-down.</li> <li>■ Priority numbers can range from 1 (highest priority) to 128 (lowest priority).</li> <li>■ The background task is not shown because it always has the lowest priority.</li> <li>■ Interrupt-driven tasks are named after the corresponding interrupt blocks.</li> </ul>



- Timer tasks are named after their sample times.  
The number of timer tasks depends on the execution mode. For details, refer to *Scheduler mode* on page 47.

---

**Up** Increases the priority of the highlighted task.

**As Previous** Assigns the priority of the previous task to the currently selected task.

**Down** Decreases the priority of the highlighted task.

**Task name** Displays the name of the currently selected task.

---

**Interrupt source** Displays the name and the CPU of the interrupt source of the currently selected task. This can be either a hardware or a software interrupt or a timer interrupt.

**Interrupt block** Displays the name and path of the hardware or software interrupt block if an interrupt block-driven task is selected.

**Priority** Displays the priority of the currently selected task.

**Stop simulation** The simulation is stopped whenever a task overrun occurs.

**Queue task before simulation stop** When a task overrun occurs, the task instance is queued for later execution.



**Max. queued task instances** Maximum number of queued task instances. If this limit is reached, the simulation is stopped.

**Ignore overrun** When a task overrun occurs, the task concerned is not scheduled. As a result, task instances are discarded in overrun situations.

---

## Related topics

### HowTos

- [How to Change the Task Priorities in RTI-MP Models](#) 
- [How to Handle Overrun Situations](#) 

### References

- [CPU's Page \(Multiprocessor Setup Dialog\)](#) on page 49
- [priority](#) on page 86

## CPU Options Dialog

Access	You can access this dialog via the <b>Build Options</b> and <b>Variable Description File Options</b> buttons on the CPU pages of the <b>Multiprocessor Setup</b> dialog.
Dialog pages	<p>This dialog contains the following pages:</p> <ul style="list-style-type: none"><li>■ <i>Build Options Page (CPU Options Dialog)</i> on page 58 for setting compiler and other build options.</li><li>■ <i>Variable Description File Options Page (CPU Options Dialog)</i> on page 61 for setting options that control the generation of the variable description file.</li></ul>

## Build Options Page (CPU Options Dialog)

Access	This page is contained in the <b>CPU Options</b> dialog. You can access it via the Build Options button on the CPU's page of the <b>Multiprocessor Setup</b> dialog.
Purpose	To set compiler and other build options.
Dialog settings	<p><b>Compiler options</b> Lets you specify the compiler options. The following options are especially important:</p> <ul style="list-style-type: none"><li>■ <code>-DFIRST_SIMSTEP_INCREASEMENT=&lt;Value&gt;</code> where <code>&lt;Value&gt;</code> is an arbitrary integer <math>\geq 0</math>. If the option is set for all the CPUs in the multiprocessor system, <code>&lt;Value&gt; * BaseSampleTime</code> is added to the first simulation step on each CPU. The time available on a CPU for the first simulation step is then as follows: <math>(\text{StepSizeMultiple} + \text{&lt;Value&gt;}) * \text{BaseSampleTime}</math></li></ul>

You can use this option to avoid task overruns in the first simulation step in the fastest timer task.



- This option is valid only if you specify it with the same first simulation step increment value for all the CPUs of the multiprocessor system.
- If you use this option, the time available for the first simulation step is  $(\text{StepSizeMultiple} + \text{<Value>}) * \text{BaseSampleTime}$  (meaning non real-time). During this first simulation step, the timing of I/O signals therefore does not match the timing of the simulation.

- **-DRTIMP\_GL\_NO\_TIMEOUT** If specified for a CPU, the timeout period for initializing the Gigalinks is set to infinity.  
 Downloading an application to the flash memory takes significantly longer than downloading it to the global memory. As a result, a timeout can occur when you download a multiprocessor application to the flash and one processor board attempts to synchronize its Gigalinks after the download while another is still busy with the download. The **-DRTIMP\_GL\_NO\_TIMEOUT** option makes the multiprocessor system retry the initialization of the Gigalinks until it is successful.



The initialization of the Gigalinks is not possible if the required connections are not available. If you did not specify the **-DRTIMP\_GL\_NO\_TIMEOUT** option, a timeout results, which indicates that, for example, some Gigalinks are not connected correctly, or the expansion box is switched off. If you specified the **-DRTIMP\_GL\_NO\_TIMEOUT** option, no timeout results. You should therefore use this option only if loading an application to the flash memory.

See also *Load to flash memory* on page 33.

**Compiler optimizations** Lets you specify the compiler optimization level.

- If “Default” is selected, the model is compiled with the platform- and compiler-specific default optimization, which is full inline expansion and the highest recommended level of optimization.
- If “User defined” is selected, you can define the optimization via the User-defined optimization edit field.

- If "None" is selected, the compilation is performed without any optimization.

**Effective Optimizations** Shows the optimization settings currently used for the selected compiler optimization level.

**User-defined optimization** Lets you specify user-defined compiler options. (Available only if compiler optimizations is set to "User-defined".)

- If empty, the model code is compiled with the default optimization setting of the compiler.
- You should not enter optimization levels higher than the default. For information about the permissible compiler options, refer to the following manuals:

Hardware	Manual
PowerPC-based (DS1005, DS1103, DS1104, MicroAutoBox)	<i>C/C++ Compiler User's Guide by Microtec</i>
x86-based (DS1006)	<a href="http://gcc.gnu.org/onlinedocs/">http://gcc.gnu.org/onlinedocs/</a>


**Make options** Lets you specify additional make options. There are only very few cases where additional make options are required.

**Custom TLC options** Lets you specify further code generation options.

**Assertion mode** RTI supports Simulink's Model Verification blocks for the real-time simulation.

To control the behavior of the Model Verification blocks in the real-time simulation, use RTI's **Assertion mode** setting. Depending on how you set this, one of the following actions is carried out if a verified signal leaves the desired range, meaning that the assertion occurs:

- "OFF": No reaction (default).
- "WARN": Issue a warning message in ControlDesk's Log Viewer.
- "STOP": Issue an error message and stop the simulation, i.e., set the simulation state to STOP.


For details on this technique, refer to *How to Use Simulink's Model Verification Blocks* . See also *Model Verification block enabling* on page 74.


**Enable Test Automation Stimulus Engine** Lets you enable test automation for the current CPU.

- If selected, you can use ControlDesk’s stimulus signal generation feature with the generated real-time application.
- If cleared, you cannot use ControlDesk’s stimulus signal generation feature with the generated real-time application (default).

**Enable real-time testing** Lets you enable or disable real-time testing for the generated real-time application.

## Variable Description File Options Page (CPU Options Dialog)

Access	This page is contained in the <b>CPU Options</b> dialog. You can access it via the Variable Description File Options button on the CPU’s page of the <b>Multiprocessor Setup</b> dialog.
Purpose	To set options that control the generation of the variable description file.
Global variable description file options frame	<div><p><b>Include mask and workspace parameters</b> Lets you make mask and workspace parameters available in the <b>variable description file</b>.</p><ul style="list-style-type: none"><li>■ If selected, mask and workspace parameters are available in the variable description file. The dependent parameters are set to read-only.</li><li>■ If cleared, mask and workspace parameters are not available in the variable description file. The dependent parameters are writable (default).</li></ul></div> <div> Using the TRC Exclusion Block in conjunction with the <b>Include mask and workspace parameters</b> option is not supported.</div>

You can change any number of dependent parameters by means of just one mask or workspace parameter. For details on this technique, see *Using Workspace and Mask Parameters* .



If you work with mask and workspace parameters, you have to consider some limitations. See *Using Workspace and Mask Parameters*.

Further restrictions:

- MATLAB must be installed on the host PC running the ControlDesk experiment.
- Make sure that masked subsystems are assigned to a single processor.
- All the workspace parameters are contained in the TRC file of the master CPU.
- Multidimensional parameters are not supported. For example, if you have a 3x4x2 Lookup Table (n-D) block, this comes out as one `LookupTableData` entry of the dimension 24x1 in the variable description file.

Related files: *Variable Description File (TRC File)* on page 109

**Include initial parameter values** To make the initial parameter values available in the variable description file.

- If selected, the initial parameter values of the model are generated into the variable description file (this does not apply to the Stateflow parameters of the State Machine Data group). This lets you carry out offline calibration for your model with CalDesk.
- If cleared, the initial parameter values of the model are not generated into the variable description file. For large models especially, this can significantly reduce the time needed for the code generation process (default).

Variable description files generated for ControlDesk-controlled Simulink simulations do not contain the initial parameter values, regardless of whether this option is selected or not.

#### Variable description file options

**Include signal labels** Lets you make the signal labels of the model available in the variable description file.

- If selected, the signal labels of the model are generated into the variable description file (default).

- If cleared, the signal labels of the model are not generated into the variable description file. For large models especially, this might significantly reduce the time needed for the code generation process.

Related files: *Variable Description File (TRC File)* on page 109

**Include virtual blocks** Lets you make the block outputs of virtual blocks (Mux, Demux, From, etc.) available in the variable description file.

- If selected, the virtual blocks are available in the variable description file (default).
- If cleared, the virtual blocks are not available in the variable description file. This might significantly reduce the time needed for the code generation process.

If you just want to have the outputs of a few virtual blocks available in the variable description file, you can also insert a unity Gain block (**Gain** value set to 1). Its output is available in the variable description file, even if you turn off the generation of virtual blocks.

Related files: *Variable Description File (TRC File)* on page 109

**Include states** Lets you make the states of blocks available in the variable description file.

- If selected, the block states are generated into the variable description file.
- If cleared, the block states are not generated into the variable description file (default).


Related files: *Variable Description File (TRC File)* on page 109

**Include derivatives** Lets you make the derivatives of blocks available in the variable description file.

- If selected, the derivatives of blocks are generated into the variable description file.
- If cleared, the derivatives of blocks are not generated into the variable description file (default).

Related files: *Variable Description File (TRC File)* on page 109

**Apply subsystem read/write permissions** Lets you apply Simulink's ReadOnly and NoReadNoWrite permissions for subsystems to the variable description file.

- If selected, the parameters of subsystems with the ReadOnly permission are set to `read-only` in the variable description file. The content of NoReadOrWrite subsystems is hidden in ControlDesk, only the subsystem itself is visible, including its mask parameters (if any).
- If cleared, the parameters of all subsystems are visible and writable in the variable description file, with the following exceptions:
  - A subsystem can be excluded via the **TRC Exclusion** block. Refer to *Excluding Subsystems from the TRC File* .
  - The parameters of masked subsystems are read-only if the **Include mask and workspace parameters** option is selected.



You can configure ControlDesk to show the contents of NoReadNoWrite subsystems.

Related files: *Variable Description File (TRC File)* on page 109

## Rename CPU Dialog

### Access

You can access this dialog via the Rename CPU button, which is located on the CPU's page of the **Multiprocessor Setup** dialog.

### Purpose

To change the name of the CPU.

### Dialog settings

**Enter new CPU name** Enter a name for the CPU.



- The CPU names used with RTI-MP must be identical to the CPU names used with ControlDesk's Platform Manager.
- The CPU names are restricted to eight alphanumeric characters.

### Related topics

#### References

- *CPU's Page (Multiprocessor Setup Dialog)* on page 49



## Multiprocessor Topology Setup Dialog

<b>Access</b>	You can access this dialog via the Multiprocessor network topology button (located on the Advanced page of the Multiprocessor Setup dialog).
<b>Purpose</b>	To specify the topology of a multiprocessor system.
<b>Description</b>	After you connect the Gigalinks and register your multiprocessor system, you can set up the network topology by clicking the <b>Read topology</b> button. This allows RTI-MP to build an application and download it correctly to your multiprocessor system. Each row in the <b>Communication Connection</b> frame represents one board.



As an alternative, you can specify a network topology before you register your multiprocessor system. This allows you to generate code for any multiprocessor topology. To do so, specify the number and names of the involved CPUs, and select their Gigalink connections. You can also click the Read from model button to get all the CPU names that are specified in the model. When the multiprocessor application is downloaded to the dSPACE hardware, the Gigalink connections formerly specified must be available.

### Dialog settings

**File will be saved to model directory** Displays the path of the model. Each time you click **OK** or **Apply**, the current settings in the Multiprocessor Topology Setup dialog are written to the <model>\_tp.m file located in the folder that also holds the model.

**Read topology** (Enabled after you registered your multiprocessor system.) Sets the number and names of the connected CPUs as well as their Gigalink connections in the Multiprocessor Topology Setup dialog according to the physical Gigalink connections.

**Read from model** Sets the CPU names in the Multiprocessor Topology Setup dialog according to the CPU names used in your current multiprocessor model. The values for Gigalink connections 0 ... 3 of all CPUs in your model are set to "NC" (not connected).

**Browse** Lets you choose any <model>\_tp.m file so that you can easily import the topology (CPU names and Gigalink connections) of an existing model.

### Communication Connection frame

**CPU Names** In this edit field, you can specify the names of the CPUs of your model.

- If you click the **Read topology** button, the CPU names are taken from the multiprocessor topology you registered with ControlDesk.
- If you click the **Read from model** button, the CPU names are taken from the current multiprocessor model.
- If you click the **Browse** button and select a different <model>\_tp.m file, the CPU names are imported from that file.



- If you want to download an application to your multiprocessor system, the CPU names used with RTI-MP must be identical to the CPU names used with ControlDesk's Platform Manager.
- The CPU names are restricted to eight alphanumeric characters.
- All CPU names must be unique.

**Gigalink 0 ... 3** Here you can specify the Gigalink number and CPU to which you want to connect the current Gigalink.



- If you click the **Read topology** button, the values are set automatically according to the physical Gigalink connections.
- If you click the **Read from model** button, the values are set to "NC" (not connected).
- If you click the **Browse** button and select a different <model>\_tp.m file, the values are imported from that file.

**Remove** Removes the corresponding CPU from the network topology. The Gigalinks that were connected to this CPU are disconnected ("NC").

**Add CPU** Adds a CPU to the network topology. All Gigalinks of the new CPU are disconnected ("NC").

**Reset Gigalinks** Resets all Gigalinks to the NC setting (not connected).

## Related topics

Basics

- [GigaLink Connection](#) 

References

- [Advanced Page \(Multiprocessor Setup Dialog\)](#) on page 54
- [Files Controlling the Build and Download Process](#) on page 95

## Model Configuration Parameters Dialogs

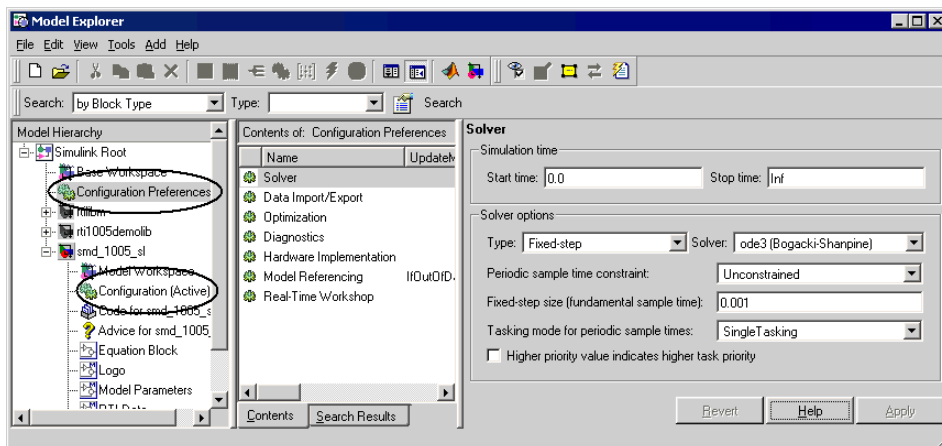
MATLAB provides configuration sets which were introduced with MATLAB R14. There are two different types of configuration sets:

- The Configuration Preferences set
- Model-specific configuration sets

**Configuration Preferences set** In the Configuration Preferences set, you can specify default values for all model properties, which are transferred to the model's configuration set when you create a new model.

**Model-specific configuration sets** Your model can have several configuration sets. You can activate a specific configuration set for a simulation or code generation.

The following illustration shows the different types of configuration sets in the Model Explorer:



**Access**

You can access the **Configuration Preferences** via the **File – Preferences** menu command of the MATLAB Desktop. Then expand the Simulink node in the Preferences tree, and click Launch model explorer. In the Model Explorer, expand the Simulink root node and select **Configuration Preferences**.

You can access the **model-specific configuration sets** by selecting **View - Model Explorer** from your model's menu bar. In the Model Explorer, expand the node of your model and select the configuration set.



You can also access the model-specific configuration set via the **Simulation - Configuration Parameters** menu command of your model.

**Description**

A few options for the program building of multiprocessor systems must be specified in the Model Explorer's Configuration Parameter dialogs. RTI-MP automatically specifies valid default settings for new models.

**Dialog pages**

The following dialog pages contain settings relevant to RTI-MP. These options are valid for the overall RTI-MP model.

- *Optimization Dialog (Model Configuration Parameters Dialogs)* on page 69 for enabling optimized code generation and defining the behavior of various blocks.
- *Diagnostics Dialog (Model Configuration Parameters Dialogs)* on page 73 for making specific model conditions generate messages.
- *Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)* on page 75 for setting certain code generation options of the RTI-MP model.

**Related topics**

HowTos

- *How to Specify Options for the Build Process* 

References

- *Multiprocessor Setup Dialog* on page 45
- *set\_rti* on page 133

## Optimization Dialog (Model Configuration Parameters Dialogs)

<b>Access</b>	This dialog is contained in the <b>Model Configuration Parameters</b> dialog. You can access it by selecting <b>View - Model Explorer</b> from your model's menu bar. In the Model Explorer, expand the node of your model and select <b>Configuration</b> . Then select <b>Optimization</b> in the Contents pane.
<b>Purpose</b>	To set options for signal storage reuse, inline parameters, optimizing block reduction, etc.
<b>Description</b>	<p>In this dialog you can specify options that improve simulation performance and the performance of code generated from your model.</p> <p>The following settings are relevant to RTI-MP. For the settings relevant to RTI, refer to <i>Optimization Dialog (Model Configuration Parameters Dialogs)</i> on page 21. For a detailed description of the Optimization dialog and its options, refer to <i>Using Simulink</i> by The MathWorks. For details on these optimization options, refer to <i>Real-Time Workshop User's Guide</i> by The MathWorks.</p>
<b>Simulation and code generation frame</b>	<p><b>Block reduction optimization</b> This option is not supported by RTI. Make sure that it is not selected.</p> <p><b>Implement logic signals as boolean (vs. double)</b> Lets you specify whether the boolean or double data type is used for logical signals. This option can be used for compatibility with models from earlier Simulink versions where logical signals are implemented as double.</p> <ul style="list-style-type: none"> <li>■ If selected, the Simulink blocks (e.g., Logical Operator) use the <b>Boolean</b> data type for logical signals (default).</li> <li>■ If cleared, the Simulink blocks in the model use the <b>double</b> data type for logical signals.</li> </ul> <p>For reasons of downward compatibility, this setting is also respected by some RTI Bit I/O blocks.</p> <p>For general information, refer to the Simulink documentation.</p>

**Signal storage reuse** Lets you control the memory allocation for block output variables.

- If selected, Real-Time Workshop does not always use a separate block output variable for each block output but attempts to assign a single buffer to several block outputs. This typically reduces memory consumption. However, reused variables are not available in the variable description file.

You can exclude individual block outputs from this optimization to make them available in the variable description file.

For details, refer to *Signal Properties Dialog* on page 78.



You can also use the **Local block outputs** option to generate block output variables as local function variables as far as possible.

- If cleared, a separate global variable is used for each block output (default).

For details, refer to *Using Simulink* by The MathWorks.

Related options: *Enable local block outputs* on page 71

Related files: *Variable Description File (TRC File)* on page 109

**Inline parameters** Lets you inline the block parameters or generate them as variables.

- If selected, the block parameters are inlined, so their concrete values are used in the generated C code, which makes them non-modifiable. This typically reduces memory consumption and execution time. As a result, no block-specific parameter is available below the **Model Root** node of the variable description file.

To generate individual parameters as tunable variables, see *Configure* below.

- If cleared, the block parameters are generated as variables, so they are modifiable during the real-time simulation (default).

Related files: *Variable Description File (TRC File)* on page 109

**Configure** Lets you make individual parameters tunable if the Inline parameters option is selected. Parameters declared as tunable parameters are available in the variable description file via the Tunable Parameters group.

For details, refer to *Model Parameters Configuration Dialog* on page 76.

Related files: *Variable Description File (TRC File)* on page 109

## Code generation - Signals frame

**Enable local block outputs** Lets you generate block output variables as local function variables as far as possible.

- If cleared, all block output variables are generated as global function variables (default).
- If selected, Real-Time Workshop attempts to generate block output variables as local function variables.

This typically reduces memory consumption or execution time. Several block output variables might not be available in the variable description file. You can therefore exclude individual block outputs from this optimization to include them in the variable description file.



This option is active only if you selected the signal storage reuse option. Refer to *Signal storage reuse* on page 70.

To enable the local block outputs for a specific CPU, you can add the `-aLocalBlockOutputs=1` option for it to the Custom TLC options edit field on the Build Options Page (CPU Options Dialog) of the *CPU Options Dialog* on page 58.

For details on this optimization option, refer to *Real-Time Workshop User's Guide* by The MathWorks.

Related files: *Variable Description File (TRC File)* on page 109 (see page *Variable Description File (TRC File)* on page 109)

**Reuse block outputs** Lets you reuse signal memory whenever possible.

- If selected, Real-Time Workshop attempts to assign a single buffer to several signals.

Reused variables are not available in the variable description file. You can therefore exclude individual signals from this optimization to make them available in the variable description file.

- If cleared, local function variables are unique (default).



This option is active only if the **Signal storage reuse** option is set to **On**.

Refer to *Signal storage reuse* on page 70.

**Ignore integer downcasts for folded expressions** Lets you ignore integer downcasts for the microprocessor architecture used.

To enforce the integer downcast for a specific CPU, you can add the `-aEnforceIntegerDowncast=1` option for it to the Custom TLC options edit field on the Build Options Page (CPU Options Dialog) of the *CPU Options Dialog* on page 58.

**Inline invariant signals** Lets you inline invariant signals.

- If selected, the invariant signals of the model are inlined.
- If cleared, invariant signals are generated as variables (default).



This option is active only if you selected the Inline parameters option.

Refer to *Inline parameters* on page 70.

Invariant signals are constant signals or signals that are directly derived from constants. Normally, they are generated as variables so that they are modifiable during the real-time simulation. However, if they are inlined, their concrete values are used in the generated C code, which makes them nonmodifiable. This typically reduces memory consumption and execution time.

Inlined signals are not available in the variable description file. You can therefore exclude individual block outputs from this optimization to include them in the variable description file.

For details on this optimization option, refer to *Real-Time Workshop User's Guide* by The MathWorks.

Related files: *Variable Description File (TRC File)* on page 109

To inline the invariant signals for a specific CPU, you can add the `-aInlineInvariantSignals=1` option for the CPU to the Custom TLC options edit field on the Build Options Page (CPU Options Dialog) of the *CPU Options Dialog* on page 58.

**Eliminate superfluous temporary variables (Expression folding)** Lets you optimize the computation time for intermediate results at block outputs and the storage of results stored in temporary variables.

**Loop unrolling threshold** Lets you make Real-Time Workshop place a sequence of individual statements in a `for` loop, if required.

See *Real-Time Workshop User's Guide* by The MathWorks.



To configure a CPU-specific loop rolling threshold, you can add the `-aRollThreshold=<positive_number>` option for the CPU to the Custom TLC options edit field on the Build Options Page (CPU Options Dialog) of the *CPU Options Dialog* on page 58.

#### Related topics

#### References

- *Optimization Dialog (Model Configuration Parameters Dialogs)* on page 21

## Diagnostics Dialog (Model Configuration Parameters Dialogs)

#### Access

This dialog is contained in the **Model Configuration Parameters** dialog. You can access it by selecting **View - Model Explorer** from your model's menu bar. In the Model Explorer, expand the node with your model's name and select **Configuration**. Then select **Diagnostics** in the Contents pane.

#### Purpose

To specify options that make specific model conditions generate either error messages or warnings.

#### Description

On this page you can specify options to check the overall RTI-MP model and the code generation process. The Diagnostics dialog contains the following pages:

- The **Solver** page lets you specify the diagnostic actions Simulink takes when it detects a solver-related error.
- The **Sample Time** page lets you specify the diagnostic actions Simulink takes when it detects an error relating to the sample times in your model.
- *Data Validity page* on page 74 lets you specify the diagnostic actions Simulink takes when it detects an error that could have an impact on the data integrity of your model.
- The **Conversion** page lets you specify the diagnostic actions Simulink takes when it detects a data type conversion error during compilation of your model.
- The **Connectivity** page lets you specify the diagnostic actions Simulink takes when it detects an error relating to block connections.

- The **Compatibility** page lets you specify the diagnostic actions Simulink takes when it detects an error relating to compatibility between the current Simulink version and your model.
- The **Model Referencing** page lets you specify the diagnostic actions Simulink takes when it detects an error relating to Model Referencing.

In this reference you will find only the settings relevant to real-time simulation. For a detailed description of the Diagnostics dialog and its options, refer to the MATLAB documentation by The MathWorks.


#### Data Validity page

**Model Verification block enabling** Lets you globally enable or disable all the Model Verification blocks or use the settings of the individual blocks.



- If “Enable all” is selected, all the **Model Verification** blocks are active.
- If “Disable all” is selected, none of the **Model Verification** blocks is active.
- If “Use local settings” is selected, only the **Model Verification blocks** with the **Enable assertion** setting selected are active (default).

To control the behavior of the Model Verification blocks in the real-time simulation, use RTI’s **Assertion mode** setting. Depending on how you set this, one of the following actions is carried out if a verified signal leaves the desired range, meaning that the assertion occurs:

- “OFF”: No reaction (default).
- “WARN”: Issue a warning message in ControlDesk’s Log Viewer.
- “STOP”: Issue an error message and stop the simulation, i.e., set the simulation state to STOP.

For details on this technique, refer to *How to Use Simulink’s Model Verification Blocks* (  *RTI and RTI-MP Implementation Guide*).




# Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)

Access	<p>This dialog is contained in the <b>Model Configuration Preferences</b> dialog. You can access it by selecting <b>View - Model Explorer</b> from your model's menu bar. In the Model Explorer, expand the node of your model and select <b>Configuration</b>. Then select <b>Real-Time Workshop</b> in the Contents pane.</p>
Purpose	<p>To set Real-Time Workshop and RTI-MP code generation options.</p>
Description	<p>In this dialog you can specify options relating to the code generation process.</p> <p>The following settings are relevant to RTI-MP. For the settings relevant to RTI, refer to <i>Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)</i> on page 27. For a detailed description of the Real-Time Workshop dialog and its options, refer to <i>Real-Time Workshop User's Guide</i> by The MathWorks.</p>
Dialog settings	<div> <div>Generate code only</div> <div>Lets you generate only C code from your Simulink model.</div> </div> <ul style="list-style-type: none"> <li>■ If selected, the model code is only generated. The code is neither compiled nor downloaded to the hardware.</li> <li>■ If cleared, the model code is generated, compiled, and downloaded to the dSPACE hardware (default).</li> </ul>
Related topics	<div>Basics</div> <ul style="list-style-type: none"> <li>• <i>Basics for the Build and Download</i> </li> <li>• <i>Building and Downloading the Model</i> </li> </ul> <div>References</div> <ul style="list-style-type: none"> <li>• <i>Model Parameters Configuration Dialog</i> on page 76</li> <li>• <i>Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)</i> on page 27</li> </ul>

## Model Parameters Configuration Dialog

<b>Access</b>	You can access this dialog via the Configure button, which is located in the Optimization dialog of the Configuration Parameters dialog in the Model Explorer. Refer to <i>Optimization Dialog (Model Configuration Parameters Dialogs)</i> on page 69.
<b>Purpose</b>	To declare specific parameters as tunable parameters.
<b>Description</b>	You can make individual parameters tunable if the Inline parameters option is selected. The fewer parameters you declare tunable, the more memory and execution time can be saved.
<b>Dialog settings</b>	This section gives only a brief summary of the dialog settings. For a detailed description of the <b>Tunable Parameters</b> dialog and its options, refer to <i>Using Simulink</i> by The MathWorks.
<b>Global (tunable) parameters frame</b>	<p><b>New</b> Adds a new tunable parameter.</p> <p><b>Remove</b> Removes a tunable parameter.</p> <p><b>Name</b> Enter the parameter name here.</p> <p><b>Storage class</b> Select the storage class for the parameter.</p> <p><b>Storage type qualifier</b> Enter the storage type qualifier here.</p>
<b>Source List frame</b>	<p>From the drop-down list, select whether you want to have all “MATLAB workspace” variables or only the “Referenced workspace variables” displayed in the source list.</p> <p>The source list displays all the workspace variables that correspond to the selection of the drop-down list. The variables that are made global (tunable) parameters are displayed in bold italics.</p> <p><b>Refresh list</b> Updates the list from the workspace/model.</p> <p><b>Add to table &gt;&gt;</b> Adds the highlighted variable(s) to the Global (tunable) parameters list.</p>
<b>Related topics</b>	<p>References</p> <ul style="list-style-type: none"> <li>• <i>Optimization Dialog (Model Configuration Parameters Dialogs)</i> on page 69</li> </ul>

## External Mode Control Panel

<b>Access</b>	You can access this dialog for a submodel via the <b>Tools – External Mode Control Panel</b> menu command of that RTI-MP submodel.
<b>Purpose</b>	To specify options for the external simulation.
<b>Description</b>	For details on the external simulation mode, refer to <i>External Simulation</i>  in the <i>RTI and RTI-MP Implementation Guide</i> .
<b>Dialog settings</b>	<p><b>Connect/Disconnect</b> Toggles the external simulation:</p> <ul style="list-style-type: none"> <li>■ <b>Connect</b> switches the Simulink simulation mode from <i>normal</i> to <i>external</i> and the target interface for external mode connects to the real-time application. After the connection is established, the entire set of the current parameter values of the Simulink model is downloaded. This ensures that the Simulink model and the corresponding real-time application use the same set of parameters.</li> </ul> <div data-bbox="511 857 1202 1022">  If parameters of the application were already altered with ControlDesk, these parameter changes are overwritten with the current parameters values of the Simulink model when you start the external simulation. </div> <ul style="list-style-type: none"> <li>■ <b>Disconnect</b> leaves the simulation set to external, but parameter changes in the current Simulink model are no longer passed to the hardware.</li> </ul> <div data-bbox="482 1171 1202 1307">  Starting and stopping the external simulation does not affect the simulation state of the real-time simulation. To start or stop the real-time simulation, you can change the <code>simState</code> variable via ControlDesk, for example. </div> <p><b>Start/Stop Real-Time Code</b> Not supported.</p> <p><b>Arm Trigger</b> Not supported. Leave it cleared.</p> <p><b>Enable data uploading</b> This checkbox works as an Arm trigger button for floating scopes. For details, refer to the MATLAB documentation.</p> <p><b>Duration</b> Lets you specify the duration for floating scopes.</p>

**Batch download** Lets you select the download mode.

- If selected, parameter changes in the Simulink model are stored and not passed on to the hardware until the **Download** button below the checkbox is clicked.
- If cleared, parameter changes in the current Simulink model are immediately passed on to the hardware.

**Download** Parameter changes that have been stored are passed on to the hardware (enabled only if Batch download is selected).

**Signal & triggering** Not supported.

**Data archiving** Not supported.

Do not change the settings that are contained in the subdialog that opens.

**Related topics**

Basics

- *External Simulation* 

HowTos

- *How to Run an External Simulation* 

References

- *Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)* on page 75

# Signal Properties Dialog

**Access**

You can access this dialog via the **Edit – Signal Properties** menu command of the **Simulink** model (available only if a signal is selected in the model).

**Purpose**

To specify the properties of the selected signal.

**Description**

You can make a signal displayable even if certain code optimization options are selected.

In this reference you will find only the settings relevant to real-time simulation. For a detailed description of the **Signal Properties** dialog and its options, refer to *Using Simulink* by The MathWorks.

The Simulink Signal Properties dialog contains the following pages:

- *Logging and accessibility* page on page 79
- *Real-Time Workshop* page on page 79

■ Documentation page (not relevant to RTI)

**Signal name**    Specify the name of the signal if desired.

Logging and accessibility  
page

**Test Point**    Select this checkbox if you want to make the signal  
global.

Real-Time Workshop page

**RTW storage class**    Select the storage class for the signal if desired.





# RTI and RTI-MP Variable Reference

The real-time application provides a number of variables that contain information about the application. You can use ControlDesk, MLIB and CLIB to read them from the real-time hardware.

Some variables show different behavior depending on where the host service is executed. Consider the `overrunQueueCount` and `state` variables of a particular task, for example:

- `overrunQueueCount` is always 0 if read in the background task. This is due to the fact that the background task has the lowest priority and is therefore executed only if no other task is queued. If `overrunQueueCount` is read from within its own task, its value is always 1 or higher because it is incremented when the task is called and decremented when the task is finished.
- `state` is always `idle` (0) if read in the background task. If `state` is read from within its own task, it is always `running` (2), and if read from any other task, it can be `running` (2), `ready` (1) or `idle` (0).

If you use one of ControlDesk's virtual instruments to access a variable, ControlDesk automatically uses the host service that is executed in the background task.

If you trace a variable via ControlDesk's data acquisition instruments, you can specify the desired host service via the Capture Settings.

The following variables are available:

**Main group** The application information and control variables can be found in the Main group of the variable description file.

- `currentTime` on page 82
- `errorNumber` on page 83
- `finalTime` on page 83

- *modelStepSize* on page 84
- *rtiAssertionMode* on page 87
- *simState* on page 88 (on MP systems only for the master CPU)

**Task Info group** The variables of each individual task can be found in the corresponding Task Info group of the variable description file.

- *overrunCheckType* on page 84
- *overrunCount* on page 85
- *overrunQueueCount* on page 85
- *overrunQueueMax* on page 86
- *priority* on page 86
- *sampleTime* on page 88
- *state* on page 89
- *taskCallCount* on page 89
- *turnaroundTime* on page 90

## currentTime

**Purpose** To show the current simulation time.

**Description** This read-only variable is incremented by the fastest timer task with the smallest sample time available in the model. Its unit is seconds.

When the application is paused, *currentTime* stops incrementing. When the application is stopped, *currentTime* is reset to zero.




The dSPACE systems also provide the time-stamping feature. However, the *currentTime* variable is calculated by Real-Time Workshop using floating-point numbers, whereas the time stamps are calculated by the dSPACE Real-Time Kernel using integer numbers. Since the precision of floating-point numbers decreases the larger the numbers become, the *currentTime* variable and the associated time stamps might differ slightly.

**Related variable** *modelStepSize* on page 84

**TRC file group** Main

**Related topics**

Basics

- *Time-Stamping and Data Acquisition* 

## errorNumber

**Purpose**

To show the type of the last error.

**Description**

The value of this read-only variable indicates the type of the last error. It shows zero if no error occurred. If it has a value other than zero, you should inspect ControlDesk's Log Viewer for the exact error message and the issuing component.

**TRC file group**

Main

**Related topics**

Basics

- *Reaction to Run-Time Errors* 

HowTos

- *How to Handle Application-Specific Errors* 

## finalTime

**Purpose**

To show the time when the simulation is terminated.

**Description**

The unit of this read-only variable is seconds.

The `finalTime` variable displays the stop time setting `inf` as `-1`.

**Corresponding options****RTI**    *Stop time* on page 19**RTI-MP**    *Stop time* on page 19**TRC file group**

Main

## modelStepSize

Purpose	To show the base sample time of the model.
Description	The unit of this read-only variable is seconds.
Corresponding options	<b>RTI</b> <i>Fixed step size (fundamental sample time)</i> on page 20 <b>RTI-MP</b> <i>Basic step size</i> on page 47, <i>Step size</i> on page 51
TRC file group	Main

## overrunCheckType

Purpose	To show the behavior of the simulation in task overrun situations.								
Description	<p>This read-only variable displays the task's overrun check type configured in the RTI Task Configuration dialog. Possible values are:</p> <table><thead><tr><th>Value</th><th>Type</th></tr></thead><tbody><tr><td>0</td><td>ignore overrun</td></tr><tr><td>1</td><td>stop simulation</td></tr><tr><td>2</td><td>queue task before simulation stop</td></tr></tbody></table>	Value	Type	0	ignore overrun	1	stop simulation	2	queue task before simulation stop
Value	Type								
0	ignore overrun								
1	stop simulation								
2	queue task before simulation stop								
Corresponding option	<b>RTI</b> <i>Ignore overrun</i> on page 40, <i>Queue task before simulation stop</i> on page 39, <i>Stop simulation</i> on page 39, <i>Max. queued task instances</i> on page 39 <b>RTI-MP</b> <i>Ignore overrun</i> on page 40, <i>Queue task before simulation stop</i> on page 39, <i>Stop simulation</i> on page 39, <i>Max. queued task instances</i> on page 39								
TRC file group	Task Info								

---

**Related topics**

HowTos

- [How to Handle Overrun Situations](#) 

References

- [overrunCount](#) on page 85
- [overrunQueueCount](#) on page 85
- [overrunQueueMax](#) on page 86
- [RTI Task Configuration Dialog](#) on page 38
- [taskCallCount](#) on page 89

## overrunCount

---

**Purpose**

To help you analyze overrun situations.

---

**Description**

The `overrunCount` variable of a task counts the total number of task overruns that occurred for it.

---

**TRC file group**

Task Info

---

**Related topics**

HowTos

- [How to Handle Overrun Situations](#) 

References

- [overrunCheckType](#) on page 84
- [overrunQueueCount](#) on page 85
- [overrunQueueMax](#) on page 86
- [RTI Task Configuration Dialog](#) on page 38
- [taskCallCount](#) on page 89

## overrunQueueCount

---

**Purpose**

To help you analyze overrun situations.

---

**Description**

The `overrunQueueCount` variable displays the number of queued task instances.

---


**TRC file group**

Task Info

---

**Related topics**

## Basics

- [Overrun Situation and Turnaround Time](#) 

## HowTos

- [How to Handle Overrun Situations](#) 

## References

- [overrunCheckType](#) on page 84
- [overrunCount](#) on page 85
- [overrunQueueMax](#) on page 86
- [RTI Task Configuration Dialog](#) on page 38
- [taskCallCount](#) on page 89

## overrunQueueMax

---

**Purpose**

To help you analyze overrun situations.

---

**Description**

The `overrunQueueMax` variable displays the number specified for **Max. queued task calls** in the RTI Task Configuration dialog.

---

**TRC file group**

Task Info

---

**Related topics**

## Basics

- [Overrun Situation and Turnaround Time](#) 

## HowTos

- [How to Handle Overrun Situations](#) 

## References

- [overrunCheckType](#) on page 84
- [overrunCount](#) on page 85
- [overrunQueueCount](#) on page 85
- [RTI Task Configuration Dialog](#) on page 38
- [taskCallCount](#) on page 89

## priority

---


**Purpose**

To show the priority of a task.

---

**Description**

This variable displays the task priority, defined in the RTI Task Configuration dialog.


Corresponding dialog	<i>RTI Task Configuration Dialog</i> on page 38 (RTI) or <i>RTI Task Configuration Dialog (Multiprocessor Setup Dialog)</i> on page 56 (RTI-MP)
TRC file group	Task Info
Related topics	<p>Basics</p> <ul style="list-style-type: none"> <li>• <i>Priorities and Task-Switching Time</i> </li> </ul> <p>References</p> <ul style="list-style-type: none"> <li>• <i>RTI Task Configuration Dialog</i> on page 38</li> <li>• <i>state</i> on page 89</li> </ul>

## rtiAssertionMode

**Purpose** To read or set the assertion mode of the application.

**Description** This variable can have one of the following states:

Value	Reaction to an Assertion Failure
0	No reaction
1	Issue a warning message in ControlDesk's Log Viewer.
2	Issue an error message and stop the simulation, i.e. set the simulation state to STOP.


For details, refer to *How to Use Simulink's Model Verification Blocks*  in the *RTI and RTI-MP Implementation Guide*.

**RTI-MP** With RTI-MP, the `rtiAssertionMode` variable is CPU-specific and available from the Main group of each submodel.

**TRC file group** Main

**Related topics**




HowTos

- *How to Use Simulink's Model Verification Blocks* 

## sampleTime

Purpose	To show the sample time of a task.
Description	This read-only variable is available only for timer tasks. Its unit is seconds. In the single timer task mode its value is the same as the <code>modelStepSize</code> ; in the multiple timer task mode its value depends on the sample time of the corresponding blocks.
TRC file group	Task Info
Related topics	References <ul style="list-style-type: none"> <li>• <i>modelStepSize</i> on page 84</li> </ul>

## simState

Purpose	To read or set the simulation state of the application.
Description	<p>This variable can take on the states STOP (0), PAUSE (1), or RUN (2).</p> <p>For details on controlling the simulation via <code>simState</code>, refer to <i>Simulation Control (RUN/STOP Mechanism)</i> .</p> <p><b>RTI-MP</b> For multiprocessor systems the simulation is controlled by a single CPU: the <i>master CPU</i> (on the Main page of the Multiprocessor Setup dialog this CPU is marked with an <b>(M)</b>). Therefore, the <code>simState</code> variable is only accessible on the master CPU.</p>
TRC file group	Main  (RTI-MP: Main group of the master CPU's TRC file)
Related topics	Basics <ul style="list-style-type: none"> <li>• <i>Simulation Control (RUN/STOP Mechanism)</i> </li> </ul> HowTos <ul style="list-style-type: none"> <li>• <i>How to Set and Read the Simulation State</i> </li> </ul>



## state

**Purpose** To show the current state of a task.

**Description** This read-only variable can take on the following task states:

State	Value	Meaning
<b>Idle</b>	0	The task is inactive, waiting to be triggered.
<b>Ready</b>	1	The task has been triggered but could not start due to a high-priority task that is currently executing. It is waiting to start the execution.
<b>Running</b>	2	The task has started its execution. This state is true until the task finishes its execution, regardless of whether it is suspended by a high-priority task or not.

Some changes of the `state` variable might not be observed by ControlDesk. For example, if the state of a task changes faster than the ControlDesk service code is executed, then state switches cannot be recorded because the service code is in another task. It is not possible to observe changes of the `state` variable of the task that is running the ControlDesk service code.

**TRC file group** Task Info

### Related topics

Basics

- *Task States and Execution Order* 


References

- *priority* on page 86

## taskCallCount

**Purpose** To help you analyze overrun situations.

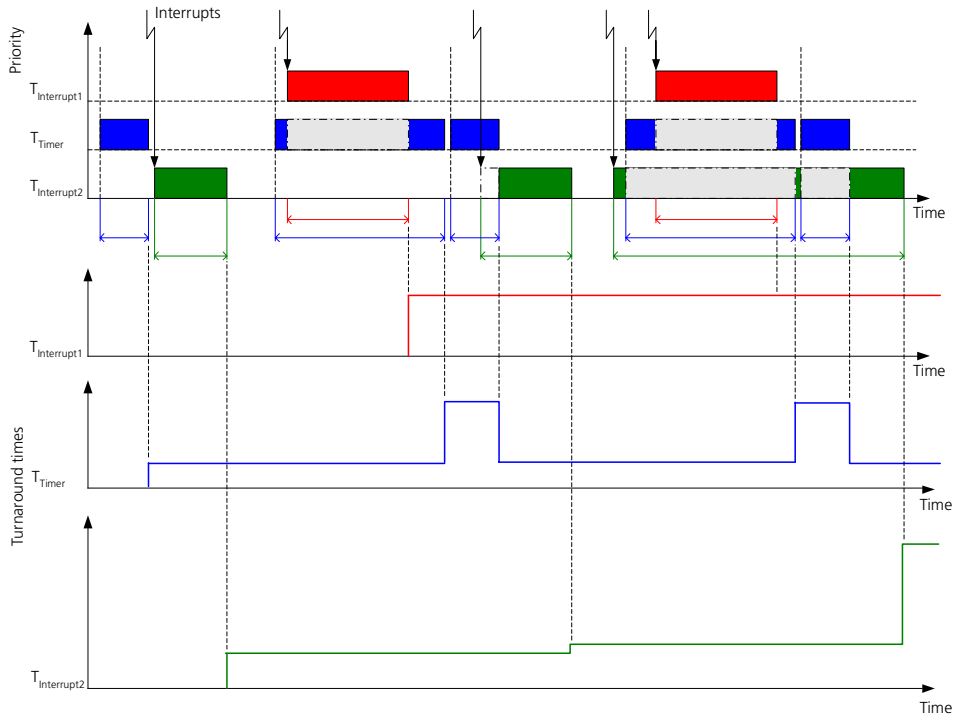
**Description** The `taskCallCount` variable of a task is incremented whenever the task is called and executed. It does not include the task calls that caused overrun situations.

TRC file group	Task Info
Related topics	<p>HowTos</p> <ul style="list-style-type: none"><li>• <i>How to Handle Overrun Situations</i> </li></ul> <p>References</p> <ul style="list-style-type: none"><li>• <i>overrunCheckType</i> on page 84</li><li>• <i>overrunCount</i> on page 85</li><li>• <i>overrunQueueCount</i> on page 85</li><li>• <i>overrunQueueMax</i> on page 86</li><li>• <i>RTI Task Configuration Dialog</i> on page 38</li></ul>

## turnaroundTime

Purpose	To show the turnaround time of a task.
Description	<p>This read-only variable shows the turnaround time, which consists of:</p> <ul style="list-style-type: none"><li>■ The time it takes to execute the functional code of the task</li><li>■ The period it takes to trace variables if the task contains the host service code (provided that ControlDesk is active)</li><li>■ The time the task's execution is interrupted by the execution of other tasks with higher priorities</li><li>■ The turnaround time does not include the task switching time.</li></ul> <p>The turnaround time is specified in seconds.</p>

The illustration below shows the turnaround times for a number of tasks.





Conditional execution paths in the model code can cause the turnaround time of a task to vary from one execution step to another. For example, if you run your model in the multiple timer task mode, the fastest task actually contains only blocks that are executed with this sample time. Nevertheless the turnaround time of this task might have periodical peaks occurring with the period of a slower task, which result from the interaction with the slower timer tasks.

- If the slower and faster tasks exchange data, the faster task contains Zero-Order Hold or Unit Delay blocks that are executed with the period of the slower task.
- To ensure that all the inputs (ADC, digital input, etc.) of the timer tasks are read almost simultaneously, parts of the code generated for RTI input blocks are always executed in the fastest task even if the RTI block is part of a task with a longer sample time.

---

**TRC file group**
**Task Info**


---

**Related topics**

## Basics

- *Overrun Situation and Turnaround Time*

---

# RTI and RTI-MP File Reference

	Several intermediate files are generated during or required by the build process. These are located in the \<[sub]model>_<target>\ build folder(s), where <target> is your active platform support. The current working folder contains user-supplied files (for example, <[sub]model>_usr.*) and the files that are needed for the real-time application (such as the real-time object, variable description file, map file, etc.).
	In addition to the files of the individual models, there are some global customization files that let you adapt the simulation environment to your needs.
<b>Overview of the model-specific files</b>	In addition to your work files (the Simulink model file, for example), there are many file types that are generated and used temporarily. For information on the files that are specific to RTI and RTI-MP, refer to <i>File Overview</i> on page 94.
<b>Files with options</b>	There are various files that can be customized. Refer to <i>File Details</i> on page 101.

## File Overview

---

Many files are created and involved in the build and download process for a model:

- Simulink files (see page *Simulink Files* on page 94)
- Files generated by RTI and RTI-MP (see page *Files Generated by RTI and RTI-MP* on page 94)
- Files controlling the build and download process (see page *Files Controlling the Build and Download Process* on page 95)
- Executable real-time object and system description files (see page *Executable Real-Time Object and System Description Files* on page 97)
- Variable access files for ControlDesk, MLIB and CLIB (see page *Variable Access Files for ControlDesk/MLIB/CLIB* on page 98)
- User-supplied files (see page *User-Supplied Files* on page 99)

## Simulink Files

---

The following file is used by Simulink.

**<[sub]model>.mdl** The MDL file contains the main Simulink model. With RTI-MP, the CPU-specific <submodel1>.mdl file contains the Simulink model for a submodel running on the specified CPU. The submodel is automatically extracted from the main RTI-MP model during the build process.

## Files Generated by RTI and RTI-MP

---

The following files are generated by Real-Time Interface.

**<[sub]model>\_th.c** This file is generated by RTI and defines the different tasks of the (sub)model. It is used to initialize the dSPACE Real-Time Kernel with the desired tasks. With RTI-MP, it is CPU-specific.

**<[sub]model>\_rti.c** This file is generated by RTI and contains definitions and declarations that are mainly related to the I/O functionality of the model or submodel. With RTI-MP, it is CPU-specific.

**<[sub]model>\_usr.c** RTI generates this template file for the User-Code if it does not already exist. You can place user-written C code in the file. The code is included in the application. With RTI-MP, it is CPU-specific.

See *User-Code File (USR.C File)* on page 103 for further information.

**<submodel>\_simeng.c** This CPU-specific file is generated by RTI-MP and controls the simulation for the corresponding submodel.

**<submodel>\_ipc.h** This CPU-specific file is generated by RTI-MP and contains all the defines and macros needed for interprocessor communication (IPC).

## Files Controlling the Build and Download Process

The following files control the build and download process.

**<model>\_tp.m** During the build process, RTI-MP maps the logical IPC channels to the network topology, which is described in this file. Whenever you change the Gigalink connections of a system, you have to update the file via RTI-MP's Multiprocessor Topology Setup dialog.

See *Multiprocessor Topology Setup Dialog* on page 65 for further information.

**rtimp\_<model>\_info.html** In each build process, RTI-MP stores information about the CPU IDs, the topology of the multiprocessor system and the mapping of the IPC channels for that topology. You might need this information for debugging because some messages contain only the IDs of the affected CPUs.

**rti[mp]<xxxx>.tlc** This is the system target file. It controls the generation of the model code and the RTI-specific files.

**rti[mp]<xxxx>.tmf** This is the template makefile. It contains all commands to compile and link a model or submodel for a certain dSPACE processor board. The platform-specific template makefile (`rti[mp]<xxxx>.tmf`) provides the basis to automatically implement a C-coded Simulink model for real-time execution. It contains all the

commands to compile and link a model or submodel for a certain dSPACE system. When generating new code for a model or submodel, Real-Time Workshop derives the application-specific makefile `<[sub]model>.mk` from the template makefile. With RTI-MP, the application-specific makefile is CPU-specific.

See also `<[sub]model>.mk` on page 96 and *User Makefile (USR.MK File)* on page 105.

**<[sub]model>.mk** Real-Time Workshop derives this application-specific makefile from the template makefile, and inserts model-specific macro variables, like the integration algorithm for the submodel and the names of the S-functions that are used. With RTI-MP, it is CPU-specific.

See also `rti[mp]<xxxx>.tmf` on page 95 and *User Makefile (USR.MK File)* on page 105.

**<[sub]model>\_rti.mk** RTI generates this model-specific include makefile. It is incorporated into the `<[sub]model>.mk` file.

**<[sub]model>\_usr.mk** This is the user makefile. It lets you modify the standard build process via a set of make macros. You can define extra search paths (for S-functions or user-defined C source code), user libraries, etc.

See *User Makefile (USR.MK File)* on page 105 for further information.

**<[sub]model>\_rti.prj** RTI uses this model-specific project marker file to determine whether the source files need to be recompiled.

**<[sub]model>.lk** The linker command file describes the assignment of memory sections to the memory banks of the corresponding processor.

See *Linker Command File (LK File)* on page 108 for further information.

**build.bat** This RTI-MP-specific batch file rebuilds a multiprocessor application. For further help on this command, enter **build /?** in a DOS window.

You should consider using the `rtimp_build` command instead (see page *rtimp\_build* on page 128).

**clean.bat** This RTI-MP-specific batch file deletes all the temporary files that are generated by RTI-MP on each build process. For further help on this command, enter **clean /?** in a DOS window.

You should consider using the `rtimp_build` command instead (see page *rtimp\_build* on page 128).



**download.bat** This RTI-MP-specific batch file downloads a multiprocessor application to the dSPACE hardware. For further help on this command, enter **download /?** in a DOS window.

You should consider using the `rtimp_build` command instead (see page *rtimp\_build* on page 128).

## Executable Real-Time Object and System Description Files

The following files are the real-time object files that are executed on your real-time hardware. In addition, there is the system description file that binds the object file(s) and variable description file(s) together for easy access in ControlDesk.

**<[sub]model>.ppc** RTI generates this file during the build process. It is the executable object file for the DS1103, DS1104 and DS1005 boards and MicroAutoBox. After the build process is finished, it can be downloaded to the dSPACE real-time hardware.

**<[sub]model>.x86** RTI generates this file during the build process. It is the executable object file for the DS1006 Processor boards. After the build process is finished, the file can be downloaded to the dSPACE real-time hardware.



**<model>.sdf** This file (called the system description file) is generated during each build process. It contains information about the CPU name(s), the corresponding object file(s) to be downloaded and the corresponding variable description file(s).



The `<model>.sdf` file is generated during each build and download process. Additional user-specific entries, such as for loading DS230x applications, must be made in the `<model>_usr.sdf` file.

See *Creating System Description Files*  for further information.

**<model>\_usr.sdf** RTI and RTI-MP support an optional user-specific system description file (`<model>_usr.sdf`), for example, for the automated download of DS230x applications via SDF files. You can create it with ControlDesk's SDF File Editor. If the file exists in the working folder, its contents are incorporated into the `<model>.sdf` file, which RTI and RTI-MP generate for the application. You can also merge the file into the `<model>.sdf` file manually using the `rti_sdfmerge` command (see page *rti\_sdfmerge* on page 122).

See *How to Download DS230x Applications*  and *Creating System Description Files*  for further information.

#### Related topics

HowTos

- *How to Update TRC and SDF Files* 


References

- *rti\_sdfmerge* on page 122


## Variable Access Files for ControlDesk/MLIB/CLIB

The following files allow you to access variables of the real-time simulation.

**<[sub]model>.trc** RTI generates this variable description file, which contains the description of all signals and parameters of the model or submodel. The signals and parameters are ordered according to the hierarchical structure of the (sub)model. The file is referenced by the SDF file of the real-time application to allow access to the signals and parameters of the application, for example, via dSPACE's ControlDesk and MLIB/MTRACE. With RTI-MP, it is CPU-specific.

See *Variable Description File (TRC File)* on page 109 for further information. For a description of the file syntax, refer to *Syntax of the TRC File* .

**<[sub]model>\_usr.trc** This is the user variable description file, which you can create yourself. If it exists, it is automatically inserted into the variable description file (<[sub]model>.trc) during the build process. It lets you specify additional variables that are available for ControlDesk. It can contain entries for internal variables of S-functions or user-defined libraries. With RTI-MP, it is CPU-specific. You can also merge the file into the <[sub]model>.trc file manually using the *rti\_usrtrcmerge* command (see page *rti\_usrtrcmerge* on page 122).

See *User Variable Description File (USR.TRC File)* on page 118 for further information. For a description of the file syntax, refer to *Syntax of the TRC File* .

**<[sub]model>.map** The linker generates this linker map file, which describes the memory layout of an application. ControlDesk and MLIB/MTRACE use it to retrieve variable addresses. With RTI-MP, it is CPU-specific.

---

**Related topics**

HowTos

- [How to Update TRC and SDF Files](#) 

References

- [rti\\_usrtrcmmerge](#) on page 122

---

## User-Supplied Files

The following files allow you to customize the real-time application and the build and download behavior.

**<model>\_tp.m** During the build process, RTI-MP maps the logical IPC channels to the network topology, which is described in this file. Whenever you change the Gigalink connections of a system, you have to update the file via RTI-MP's Multiprocessor Topology Setup dialog.

See *Multiprocessor Topology Setup Dialog* on page 65 for further information.



**<[sub]model>\_usr.c** RTI generates this template file for the User-Code if it does not already exist. You can place user-written C code in the file. The code is included in the application. With RTI-MP, it is CPU-specific.

See *User-Code File (USR.C File)* on page 103 for further information.


**<[sub]model>\_usr.mk** This is the user makefile. It lets you modify the standard build process via a set of make macros. You can define extra search paths (for S-functions or user-defined C source code), user libraries, etc.


See *User Makefile (USR.MK File)* on page 105 for further information.

**<model>\_usr.sdf** RTI and RTI-MP support an optional user-specific system description file (<model>\_usr.sdf), for example, for the automated download of DS230x applications via SDF files. You can create it with ControlDesk's SDF File Editor. If the file exists in the working folder, its contents are incorporated into the <model>.sdf file, which RTI and RTI-MP generate for the application. You can also merge the file into the <model>.sdf file manually using the `rti_sdfmerge` command (see page [rti\\_sdfmerge](#) on page 122).

See *How to Download DS230x Applications*  and *Creating System Description Files*  for further information.



**<[sub]model>\_usr.trc** This is the user variable description file, which you can create yourself. If it exists, it is automatically inserted into the variable description file (<[sub]model>.trc) during the build process. It lets you specify additional variables that are available for ControlDesk. It can contain entries for internal variables of S-functions or user-defined libraries. With RTI-MP, it is CPU-specific. You can also merge the file into the <[sub]model>.trc file manually using the `rti_usrtrcmerge` command (see page [rti\\_usrtrcmerge](#) on page 122).

See *User Variable Description File (USR.TRC File)* on page 118 for further information. For a description of the file syntax, refer to *Syntax of the TRC File* .

**<[sub]model>.ddc** This model-specific file contains information on loading a DS230x application. When RTI finds this file during a build process, it generates the necessary function calls, and compiles and links the DDC file. You can generate this file for a Simulink model as described in *How to Download DS230x Applications* .

## Related topics

### HowTos

- [How to Download DS230x Applications](#) 
- [How to Update TRC and SDF Files](#) 

### References

- [rti\\_sdfmerge](#) on page 122
- [rti\\_usrtrcmerge](#) on page 122

# File Details

The following files have settable options:	
Customizing the simulation environment	<ul style="list-style-type: none"> <li>■ <i>startup.m</i> on page 101</li> <li>■ <i>dsstartup.m</i> on page 102</li> <li>■ <i>dsfinish.m</i> on page 103</li> </ul>
Inserting custom C code	<ul style="list-style-type: none"> <li>■ <i>User-Code File (USR.C File)</i> on page 103</li> </ul>
Make process	<ul style="list-style-type: none"> <li>■ <i>User Makefile (USR.MK File)</i> on page 105</li> <li>■ <i>Linker Command File (LK File)</i> on page 108</li> <li>■ <i>User System Description File (USR.SDF File)</i> on page 108</li> </ul>
Variable access in ControlDesk/MLIB/CLIB	<ul style="list-style-type: none"> <li>■ <i>Variable Description File (TRC File)</i> on page 109               <ul style="list-style-type: none"> <li>■ <i>Variable Description File Groups</i> on page 113</li> <li>■ <i>Available Variables in the Variable Description File</i> on page 115</li> <li>■ <i>Data Type Specifications</i> on page 118</li> </ul> </li> <li>■ <i>User Variable Description File (USR.TRC File)</i> on page 118</li> </ul>

## startup.m

Purpose	To carry out user-specific commands before the dSPACE software is initialized.
Result	Each time you start MATLAB or activate a different RTI platform support, this M file is executed before the dSPACE software is initialized.
Description	Place this M file in the MATLAB search path, for example, in MATLAB's working folder. All valid MATLAB commands are allowed in it. Use the standard M file syntax.




To carry out dSPACE-specific commands, you must use the *dsstartup.m* file.

The `startup.m` file is a standard MATLAB feature. For further information type **doc startup** at the MATLAB prompt.

#### Related topics

HowTos

- *How to Customize the MATLAB Start-Up* 

References

- *dsstartup.m* on page 102

## dsstartup.m

#### Purpose

To carry out user-specific commands after the dSPACE software is initialized.

#### Result

Each time you start MATLAB or switch to a different platform, this M file is executed after the dSPACE software is initialized.

#### Description

Place this M file in the MATLAB search path, for example, in MATLAB's working folder. All valid MATLAB commands are allowed in it, including the dSPACE-specific commands. Use the standard M file syntax.

#### Example

Suppose you want MATLAB to automatically change to your working folder `c:\work` and open the Simulink model `my_model.mdl`. Write the following `dsstartup.m` file:

```
cd c:\work
my_model
```

#### Related topics

Basics

- *dsfinish.m* on page 103


HowTos

- *How to Customize the MATLAB Start-Up* 


References

- *startup.m* on page 101


# dsfinish.m

Purpose	To carry out user-specific commands before MATLAB is shut down.
Result	Each time you exit MATLAB, this M file is executed.
Description	<div> <div>Place this M file in the MATLAB search path, for example, in MATLAB's working folder. All valid MATLAB commands are allowed in it, including the dSPACE-specific commands. Use the standard M file syntax.</div> <div> <div></div> <div> <p>You can specify several dsfinish.m files. All dsfinish.m files located in the MATLAB search path are executed before MATLAB is shutdown.</p> <p>Available user-specific finish.m files are obsolete and are not executed since dSPACE software implements its own finish.m file.</p> </div> </div> </div>

## Related topics

- HowTos
- How to Customize the MATLAB Start-Up 
- References
- dsstartup.m on page 102
  - startup.m on page 101

# User-Code File (USR.C File)

Purpose	To add hand-written C code to the real-time application.
Description	<div>The User-Code file &lt;[sub]model&gt;_usr.c is included in both the initialization part and the evaluation sequence of the timer task that is executed at the base sample rate. Therefore, you can incorporate access to I/O devices into a real-time program via the User-Code file. For information on how to write a User-Code file and transfer data from and to the Simulink model, refer to <i>Implementing User-Code</i> .</div>

RTI generates a model-specific template User-Code file during the first build process for the corresponding Simulink model. If this file already exists, no new template file is generated. The template User-Code file contains the following C functions, which are automatically called by RTI:

**usr\_initialize()** The user-specific code that you want to be executed during program initialization, for example, user-specific initialization of I/O devices, must be placed within this function. It is called after the model initialization function `MdlStart()`, which originates from the Simulink model.

**usr\_sample\_input()** The user-specific code that you want to use for sampling input devices (start converters and read digital input devices) must be placed within this function. In the RUN mode it is executed before the inputs of the RTI input blocks are sampled.

**usr\_input()** This function also holds function calls for reading input devices. In the RUN mode it is executed after the inputs of the RTI input blocks are sampled but before the model code of the Simulink model is executed. It can also be used for special purposes such as overwriting values already read by RTI.

**usr\_output()** This function holds function calls for writing to output devices. In the RUN mode it is executed after the model code of the Simulink model.

**usr\_terminate()** The code to be executed when the real-time application terminates must be placed within this function, for example, to write particular termination values to the output devices whenever the simulation is stopped. In the real-time application, it is executed whenever the `simState` variable is changed to STOP mode. The function is executed after the model termination function `MdlTerminate()`, which originates from the Simulink model.

**usr\_background()** If your real-time application has to execute some code in the background task, place the code within this function.

---

## Related topics

### Basics

- *Execution Order of S-Functions and User-Code* 
- *Implementing User-Code* 

### References

- *simState* on page 88



## User Makefile (USR.MK File)

**Purpose** To define extra search paths (for S-functions or user-defined C source code), user libraries, etc.

**Description** The user makefile lets you modify the standard build process via a set of make macros. The following paragraphs list the different options, which you can find in the <[sub]model>\_usr.mk file under the headline Macros for user-specific settings.



- You can use the backslash character \ as a line-continuation character.
- Do not indent the options via spaces or tabs. However, you may indent the parameters of an option if they are continued in a new line. For example:

```
SFCN_DIR = \project1\sfcns \
          \project2\sfcns
```

- If path names contain whitespaces they need to be set in double quotes. For example:

```
SFCN_DIR = "\project one\sfcns" \
          "\project two\sfcns"
```

Note that whitespaces and double quotes with path names are available since RTI from dSPACE Release 6.0. Such paths do not work with previous versions of RTI.

- Whitespaces and double quotes are not supported with file names like "my source.c".



You can use environment variables or make macros when specifying file or path names, for example, in the user makefile you could write

```
USER_LIBS = $(PROJECTPATH)\libraries\mylib.lib
```

If you do not want to make PROJECTPATH an environment variable, you can specify it as a make macro, for example:

- RTI : make\_rti PROJECTPATH="C:\project1"
- RTI-MP : Make options: PROJECTPATH="C:\project1"

**SFCN\_DIR** These folders are used as the search path for the source code of S-functions. As a result you can collect the source code of frequently used S-functions in these common folders and do not need to keep it in the current working folder. For example:

```
# Directories where S-Function C source files are stored.
SFCN_DIR = \project1\sfcns \
           \project2\sfcns
```



If you also want to place the corresponding MEX DLL files in the S-function folder, you have to add it to the MATLAB search path as well.

**USER\_SRCS** The additional C source code files are compiled and linked. These files need to have the extension `.c`, which marks them as C source code files. They have to be located in the working folder of your model or on the search path that is specified via the `USER_SRCS_DIR = <...>` make macro. For example:

```
# Additional C source files to be compiled (file name extension .c).
USER_SRCS = mysource1.c mysource2.c
```



You do not need to specify S-function source files that are used in the Simulink model because their names are incorporated automatically in the list of source files to be compiled.

**USER\_ASM\_SRCS** The additional assembler source files are assembled and linked. They need to have the extension `.asm`, which marks them as assembler source code files. They have to be located in the working folder of your model or on the search path that is specified via the `USER_SRCS_DIR = <...>` make macro. For example:

```
# Add. assembler source files to be compiled (file ext .asm).
USER_ASM_SRCS = mysource1.asm mysource2.asm
```

**USER\_SRCS\_DIR** These folders are used as the search path for the C and assembler source code files that are declared via the `USER_SRCS = <...>` and `USER_ASM_SRCS = <...>` make macros. For example:

```
# Directories where add. C and assembler source files are stored.
USER_SRCS_DIR = \project1\usr \
                \project2\usr
```

**USER\_INCLUDES\_PATH** These folders are used as the search path for include files that are used in the source code of S-functions or User-Code. For example:

```
# Path names for user include files.
USER_INCLUDES_PATH = \project1\sfens \
                    \project1\usr
```

**USER\_OBJS** The precompiled object files are used to supply functions that you can use in the source code of S-functions and User-Code. For example:

```
# Additional user object files to be linked.
USR_OBJS = module1.obj module2.obj
```

The file name extension can vary and depends on the platform (for example, o40 or o05).

**USER\_LIBS** These libraries are used to supply functions that you can include in the source code of S-functions and User-Code. For example:

```
# Additional user libraries to be linked.
USR_LIBS = \project1\libs\usr1.lib \
          \project1\libs\usr2.lib
```

**Dependencies** It is quite common for user source files to depend on other files; for example, a `source.c` file needs to be recompiled if it includes a `header.h` file via the `#include` directive and that header file is modified. You can include such dependencies (even to several files) in the user makefile, for example:

```
source.c : header.h
source2.c : header1.h header2.h
Ensure you include blanks before and after each colon.
```

---

#### Related files

*rti[mp]<xxx>.tmf* on page 95, *<[sub]model>.mk* on page 96

---

#### Related topics

Basics

- *Tips and Tricks for Custom C Code* 



## Linker Command File (LK File)

<b>Purpose</b>	To provide the linker program with the necessary information on where to place the various code sections in the memory of the processor/controller board.
<b>Description</b>	<p>The linker command file is specially adapted to the needs of Real-Time Workshop/RTI-built programs so that the placement of the different sections and the sizing of the program heap are optimal for a wide range of different applications. For some applications it is desirable to work with a modified local copy of the original linker command file. As a linker command file is tightly coupled to a specific application, the naming of a local copy must be <code>&lt;model&gt;.lk</code> or <code>&lt;submodel&gt;.lk</code>. Depending on your processor/controller board, you can use one of the following files as a starting-point for your application-specific linker command file:</p>




Processor / Controller Board	Corresponding Linker Command File
DS1005	%DSPACE_ROOT%\DS1005\RTLib\ds1005.lk
DS1006	%DSPACE_ROOT%\DS1006\RTLib\ds1006.lk
DS1103	%DSPACE_ROOT%\DS1103\ds1103.lk
DS1104	%DSPACE_ROOT%\DS1104\RTLib\ds1104.lk
MicroAutoBox	%DSPACE_ROOT%\DS1401\RTLib\ds1401.lk

## User System Description File (USR.SDF File)

<b>Purpose</b>	To customize the download procedure.
<b>Description</b>	<p>The user system description file allows you to include in the download process object files that are not connected to the main application running on a dSPACE processor board, for example, DS230x applications. Whenever you build the model, RTI and RTI-MP incorporate the contents of the user system description file in the generated system description file (<code>&lt;model&gt;.sdf</code>). You can also merge the file into the <code>&lt;model&gt;.sdf</code> file manually using the <code>rti_sdfmerge</code> command (see page <i>rti_sdfmerge</i> on page 122).</p>

For details on generating user system description files, see *Creating System Description Files* . For details on downloading DS230x applications via USR.SDF files, see *How to Download DS230x Applications* .

Related topics

- Basics
- *Creating System Description Files* 
- HowTos
- *How to Download DS230x Applications* 
  - *How to Update TRC and SDF Files* 
- References
- *rti\_sdfmerge* on page 122

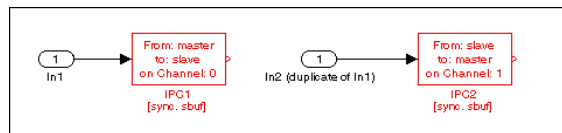
# Variable Description File (TRC File)

Purpose	To provide ControlDesk and MLIB/MTRACE with the necessary information about the signals and parameters of the application, which allows you to observe and manipulate the signals and parameters of the simulation.
Description	The structure of the variable description file reflects the hierarchy of the underlying Simulink model. Optionally, they can contain entries required by ControlDesk Test Automation. TRC files generated by RTI-MP for multiprocessor systems also support the distributed tracing.

When you build a model, RTI and RTI-MP automatically generate the necessary variable description file(s) `<[sub]model>.trc`. You can directly alter the generation of this file by various options. For details, refer to *RTI variable description file options page* on page 34 (RTI) or *Variable Description File Options Page (CPU Options Dialog)* on page 61 (RTI-MP).



- RTI and RTI-MP also provide a system description file (SDF) that bundles all the TRC files and additional information for the application. You should use the SDF file to access the signals and parameters of the application; for example, via ControlDesk.
- The variables of custom C code are not automatically included in the variable description file. To access them via ControlDesk, MLIB or CLIB you have to add them manually to the user variable description file (see *User Variable Description File (USR.TRC File)* on page 118 for details).
- TRC files generated with RTI 3.6 or higher are block-based. They contain one group with all variables (outputs, parameters, etc.) for each Simulink block in a model.
- TRC files generated with RTI 3.5.x or lower are subsystem-based. They contain one group with all variables for each subsystem in the model. If you rebuild the corresponding application with RTI 3.6 or higher, all data connections drawn from the TRC file to ControlDesk instruments are lost.
- For S-functions, TRC files do not support variables other than outputs and parameters. Use the USR.TRC file instead (see *User Variable Description File (USR.TRC File)* on page 118).
- For the blocks of the Fixed-Point Blockset, the TRC file contains scaling information for uniform fixed-point variables.
- Signal generators and viewers defined with the Signals & Scope Manager of MATLAB R14.x and later are not generated into the TRC file.
- In the TRC file, duplicate input ports are displayed as common input ports. RTI-MP checks if the duplicate input ports are modeled correctly. For example, specifying the same input signal with duplicate input ports as coming from different CPUs is not permitted.






- MATLAB R14.x and later provide Stateflow test points. A Stateflow test point is a signal that is observable during simulation. The signals's observability is assured by Stateflow. RTI generates read-only entries in the appropriate Stateflow chart group for these signals.

### Aspects of the variable description file

The following aspects are important for using the variable description file:

- The variable description file contains various groups that serve different purposes (see *Variable Description File Groups* on page 113).
- Whether a variable is available in the variable description file depends on certain conditions (see *Available Variables in the Variable Description File* on page 115).
- Depending on the hardware used, the data types in the variable description file might differ from those defined in the Simulink model (see *Data Type Specifications* on page 118).
- For information on how to load a variable description file in ControlDesk (for example, to build an instrument panel), see *How to Load System Description Files* .
- For details on the options that control the generation of TRC files, see *RTI variable description file options page* on page 34 (RTI) or *Variable Description File Options Page (CPU Options Dialog)* on page 61 (RTI-MP).

### Related files


*User Variable Description File (USR.TRC File)* on page 118

### Related topics

#### Basics

- *Excluding Subsystems from the TRC File* 

#### HowTos

- *How to Exclude a Subsystem from the TRC File* 

#### References

- *TRC Exclusion Block* on page 170
- *User Variable Description File (USR.TRC File)* on page 118
- *Variable Description File Options Page (CPU Options Dialog)* on page 61



## Variable Description File Groups

The groups of the variable description file contain various signals and parameters. The following paragraphs list the different groups of the variable description file together with a short description.


**Main** This group has the name of the (sub)model. It contains the simulation control variables, which allow online observation and manipulation of the simulation. The following simulation control variables are available:

- *currentTime* on page 82
- *errorNumber* on page 83
- *finalTime* on page 83
- *modelStepSize* on page 84
- *rtiAssertionMode* on page 87
- *simState* on page 88 (on MP systems only for the master CPU)

**Model Root** Contains the block and subsystem groups and the labeled signals for the top level of the Simulink model. Variables from the contained blocks and subsystems – including the mask parameters of masked subsystems – are located in lower hierarchical levels.

- Each subsystem group contains the labels and mask parameters defined in the subsystem, and the block groups for the contained blocks.
- Each block group contains the variables of the block, for example, outputs, parameters and states.
- Block groups for Stateflow® charts contain only the outputs to Simulink, but no states, no parameters and no local Stateflow chart data.

All Stateflow charts together form the state machine, which can have global data. This is available via the *State Machine Data* on page 114 group.

**Model Root Parameters** Contains the workspace parameters of the model. These are defined in the MATLAB workspace and referenced by the Simulink model (refer to *Include mask and workspace parameters* on page 34 (RTI) or on page *Include mask and workspace parameters* on page 34 (RTI-MP) and *Using Workspace and Mask Parameters*  in the *ControlDesk Experiment Guide* for details).

**Labels** Contains all labeled signals and appears only if labeled signals are found in the Simulink model. In this group, labels are treated as globals, and the model hierarchy is not taken into account to give quick access to important signals. The labeled signals are also available in the subsystem groups from which they originate.

**Data Stores** Collects all variables created by Data Store Memory blocks. It is created only if the Simulink model contains Data Store Memory blocks.

**User Variables from <[sub]model>\_usr.trc** Is filled with the user-specific contents of the user variable description file) <[sub]model>\_usr.trc whenever code is generated from the Simulink model. It is created only if a user variable description file is available. See also *User Variable Description File (USR.TRC File)* on page 118.

**Task Info** Contains a separate subgroup for each task in the model. The subgroups are named after the corresponding task or hardware/software interrupt block, for example, Timer Task 1, CrankEvent. Each subgroup contains the following variables:


- *overrunCheckType* on page 84
- *overrunCount* on page 85
- *overrunQueueCount* on page 85
- *overrunQueueMax* on page 86
- *priority* on page 86
- *sampleTime* on page 88
- *state* on page 89
- *taskCallCount* on page 89
- *turnaroundTime* on page 90

**State Machine Data** All the Stateflow charts of a model form the state machine. The State Machine Data group collects all data that is available in the Stateflow data dictionary and parented by the state machine. This group is only created if the model contains Stateflow charts. The outputs of the individual Stateflow charts are available via the Model Root group. The states, parameters and local Stateflow chart data are not available.

**Tunable Parameters** Collects the tunable parameters of the model. It is created only if the **Inline parameters** checkbox on the Advanced page of the Simulation Parameters dialog is selected. Refer to *Model Parameters Configuration Dialog* on page 37 (RTI) or on page *Model Parameters Configuration Dialog* on page 76 (RTI-MP).

---

**Related topics****Basics**

- *Available Variables in the Variable Description File* on page 115
- *Using Workspace and Mask Parameters* 

**References**

- *RTI and RTI-MP Variable Reference* on page 81

---

## Available Variables in the Variable Description File

Not all variables of a Simulink model are always available in the variable description file. The following criteria define whether a variable is included.

---

**Rules for dSPACE blocks**

The following rules apply to the dSPACE blocks:

- Since the parameters of most I/O blocks cannot be altered during the simulation, the groups for non-tunable I/O blocks are empty or missing completely.
- For performance reasons, RTI usually generates no code for unused I/O block channels. These are channels that are unconnected or connected only to Ground, Terminator, Inport, Outport, Goto or From blocks (in Simulink terminology, all these blocks are *virtual*). As a result, these channels are not available in the variable description file or are marked with the description “No data (unused channel)”.
- The hardware interrupt blocks, Software Interrupt block, Data Capture block, Background block and IPI block are completely virtual and offer no signal to be traced. Therefore, such blocks do not appear in the variable description file.

---

**Rules for virtual Simulink blocks**

By default, the virtual blocks are available in the variable description file. If you want to exclude them from it, you can clear the **Include virtual blocks** option. Refer to *Include virtual blocks in the Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)* on page 27 (RTI) or on the *Variable Description File Options Page (CPU Options Dialog)* on page 61 (RTI-MP).

Virtual blocks and subsystems do not have their own block output variables in the generated C code. Instead, their block output variables usually are located at different memory addresses. Since ControlDesk can handle only vector variables that are contiguous in the memory of the simulation platform, the variable description file holds the block output variables of virtual blocks and subsystems divided into sections in such a way that the single elements are contiguous. In ControlDesk's Variable Browser, the parts of such a signal are represented by SubArray variables that are similar to:

`Out{SubArray1}, Out{SubArray2}, ...`



- Parts of the virtual block output signal do not appear in the variable description file if they
  - Are not accessible (for example, data that is not a global variable or reused by several blocks)
  - Come from ground
  - Come from states
  - Are unconnected (not connected to a non-virtual block)
- Matrix sections are not fully supported: NxM matrices are treated as vectors with N\*M elements.



If you want to have the outputs of a virtual block available in the variable description file without them being divided into subarrays, you can insert a unity Gain block (Gain value set to "1"). Its output is available as one vector.

### Rules for n-dimensional look-up tables

For the Lookup Table (n-D), Direct Lookup Table (n-D), and Interpolation (n-D) using PreLookup blocks, RTI generates one or more `LookUpTableData` entries into the variable description file:

- $n = 1$ : one `LookUpTableData` vector
- $n = 2$ : one 2-dimensional `LookUpTableData` array
- $n > 2$ : several 2-dimensional `LookUpTableData` arrays (table slices), which as a whole represent the multidimensional parameter array of the look-up block.

For details, refer to *Tuning Parameters of n-D Lookup Table Blocks* .

---




**Rules for optimization methods**

There are a number of code optimization methods that usually make signals and parameters unavailable in the variable description file:

- With the Signal storage reuse option selected, block output variables may be reused by several blocks. Refer to *Signal storage reuse* in the *Optimization Dialog (Model Configuration Parameters Dialogs)* on page 21 (RTI) or in the *Optimization Dialog (Model Configuration Parameters Dialogs)* on page 69 (RTI-MP).
- With local block outputs enabled, output variables may be created as local function variables, and not as global variables. Refer to *Enable local block outputs* on page 71.
- With inlining of parameters enabled, block parameters are not available as variables in the code. Refer to *Inline parameters* in the *Optimization Dialog (Model Configuration Parameters Dialogs)* on page 21 (RTI) or in the *Optimization Dialog (Model Configuration Parameters Dialogs)* on page 69 (RTI-MP).
- With inlining of invariant signals enabled, constants or signals that depend only on constants are not available as variables in the code. Refer to *Inline invariant signals* on page 72.

---

**Rules for mask and workspace parameters**

Mask parameters and the Model Root Parameters group are only available if you select the Include mask and workspace parameters option. Refer to *Include mask and workspace parameters* in the *Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)* on page 27 (RTI) or on the *Variable Description File Options Page (CPU Options Dialog)* on page 61 (RTI-MP). For details on using mask and workspace parameters with the real-time simulation, see *Using Workspace and Mask Parameters* , *How to Use Mask and Workspace Parameters*  and *Example of Using Mask and Workspace Parameters*  in the *ControlDesk Experiment Guide*.

---

**Rules for complex variables**

Real-Time Workshop-generated code stores the real and imaginary parts of complex variables separately. Therefore, the display of the variable dimensions in ControlDesk is different from the one displayed in MATLAB or Simulink: Twice as many rows or columns are displayed in comparison to the Simulink model.

---

**Related topics****Basics**

- *Using Workspace and Mask Parameters* 

**References**

- *Optimization Dialog (Model Configuration Parameters Dialogs)* on page 21
- *Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)* on page 27
- *Variable Description File Options Page (CPU Options Dialog)* on page 61

---

## Data Type Specifications

Integer data types on the real-time hardware have the lengths specified in the Simulink model. `boolean` is a `uint8`, `double` is a 64-bit floating-point type and `single` is a 32-bit floating-point variable.

---

## User Variable Description File (USR.TRC File)


---


**Purpose**

To add variables of custom code to the variable description file, which makes them available for data capture and modification in the simulation.

---

**Description**

When writing a user variable description file, follow the syntax defined for the variable description file (refer to *Syntax of the TRC File* ) , and name it `<[sub]model>_usr.trc`. During the build process, the file is inserted into the variable description file. It must be created before the build process is started. It has to be located in the working folder of the model. You can also merge the file into the `<[sub]model>.trc` file manually using the `rti_usrtrcmmerge` command (see page *rti\_usrtrcmmerge* on page 122).

For instructions on how to create a variable description file, refer to *How to Access Custom Variables via ControlDesk* .




In ControlDesk you can only access variables that are declared as global and non-static in the C source code.



---

**Related topics**


## Basics

- *Limitations for Custom C Code* 

## HowTos

- *How to Access Custom Variables via ControlDesk* 
- *How to Update TRC and SDF Files* 

## References

- *rti\_usrtrcmerge* on page 122
- *Syntax of the TRC File* 
- *Variable Description File (TRC File)* on page 109





# RTI and RTI-MP Command Reference

RTI and RTI-MP provide some commands that allow you to manage the RTI environment and your simulation models. You can use these commands from the MATLAB Command Window or in your M files.

Purpose	Refer to
Changing RTI's platform support and migrating a model to a platform	
To change to a different platform support.	<i>rti1xxx</i> on page 126
To display the currently installed RTI version.	<i>rtiver</i> on page 127
To migrate a model to a simulation environment without RTI.	<i>rti_mdldcleanup</i> on page 137
To migrate a model to a specific RTI platform support.	<i>set_rti</i> on page 133
Updating an existing real-time application	
To update a TRC file.	<i>rti_usrtrcmmerge</i> on page 122
To update a system description file.	<i>rti_sdfmerge</i> on page 122
Building and downloading a model	
To specify default options for the code generation.	<i>rti_option TaskTransitionCheck</i> on page 138
To set options supported by the RTI options pages.	<i>rti_optionset</i> on page 133
To get options supported by the RTI options pages.	<i>rti_optionget</i> on page 135
To start an automated RTI build procedure.	<i>rti_build</i> on page 122

Purpose	Refer to
To start an automated RTI-MP build procedure.	<i>rtimp_build</i> on page 128
Accessing blocks and retrieving information	
To return the name of the CPU which a given block belongs to.	<i>rtimp_blktargetcpunameget</i> on page 127


## rti\_usrtrcmerge

**Syntax** `rti_usrtrcmerge <model>.mdl`

**Purpose** To update a TRC file with the contents of the <model>\_usr.trc file of the model without rebuilding the entire model.

**Related topics**

HowTos

- *How to Update TRC and SDF Files* 


## rti\_sdfmerge

**Syntax** `rti_sdfmerge <model>.mdl`

**Purpose** To update an SDF file with the contents of the <model>\_usr.sdf file of the model without rebuilding the entire model.

**Related topics**

HowTos

- *How to Update TRC and SDF Files* 

## rti\_build

**Syntax** `[errorFlag, errorMsg] = rti_build mdlName,  
['Command', <commands> [, <Parameter>, <Value> [, ...]]]`

**Purpose** To start the RTI build procedure for a given application.

**Description**

The command has the following outputs:

**errorFlag** 0 if no error occurred, otherwise a positive integer value.

**errorMsg** Empty string if no error occurred, otherwise a string containing information about the error.

The command has the following inputs:

**mdlName** (String) The name of the model. The model to be built must be open.

**<commands>** (String) The action to be carried out. Depending on this command an automated build, backup or cleanup procedure is carried out. Each of the possible commands has specific optional parameter-value pairs. See below for details on the commands and the related parameter-value pairs.

**<Parameter>, <Value>** (String) Options for the given command, which are specified by parameter-value pairs. See below for details on the commands and the related parameter-value pairs.

If you do not specify the 'Command', <commands> setting, `rti_build` defaults to `CodeGen&Make&Load` of the build procedure.

- Build procedure: This is carried out in three phases: code generation, make and load. You can start each phase individually or combined with the others. The following values are valid for the 'Command' to start the build procedure:

Value	Description
'CodeGen' or 'C'	Generates the real-time code for the model.
'Make' or 'M'	Compiles and links the real-time code to the real-time application. CodeGen must be invoked beforehand.
'Load' or 'L'	Loads the real-time application to the hardware. Make must be invoked beforehand.
'CodeGen&Make' or 'CM'	Combines CodeGen and Make.
'Make&Load' or 'ML'	Combines Make and Load. CodeGen must be invoked beforehand.
'CodeGen&Make&Load' or 'CML'	Combines CodeGen, Make and Load.

The following parameter-value pairs are valid for the build procedure:

- **SkipTaskConfiguration** To select whether the Task Configuration dialog is opened interactively to let you confirm any changed tasks of the model, for example, newly added interrupt blocks:

Value	Description
'on'	Skips the interactive Task Configuration during the build process (default).
'off'	Opens the Task Configuration dialog if necessary and stops the build process.



When you add a new task to your model, the task automatically has a priority. However, this priority might not comply with your needs. Therefore, to avoid an unsuitable real-time application, you should check the task configuration of the model before you start the automated build procedure, especially if you select the `SkipTaskConfiguration`, 'on' setting.

- **LogOutput** To select whether the output from the MATLAB Command Window is saved to a text file:

Value	Description
'on'	Saves the output from the MATLAB Command Window to the <code>&lt;model&gt;_rti&lt;xxxx&gt;_log.txt</code> file.
'off'	Does not save the output from the MATLAB Command window (default).

- **MakeOptions** To specify additional options for the make process which are appended to the existing make options:

Value	Description
' '	No additional make options (default).
'CC_OPTS'	Compiler options, for example, 'CC_OPTS="-H" '.
'CC_OPT_OPTS'	Alternative compiler optimization, for example, 'CC_OPT_OPTS="-O1" '.



You can also combine the 'CC\_OPTS' and 'CC\_OPT\_OPTS' settings in one value string. For example,

'CC\_OPTS="-H" CC\_OPT\_OPTS="-O1" '

- **Backup procedure:** The following value is valid for the 'Command' parameter to start the backup procedure:

Value	Description
'BackUp'	Creates a backup copy of the generated real-time application files that are necessary to load the application with ControlDesk.

The following parameter-value pairs are valid for the backup procedure:

- **BackUpFolder** To specify the location where the backup is saved. Make sure that the specified location exists before you start the backup procedure:


Value	Description
'<location>'	Saves the backup copy of the generated real-time application to this folder.

- Cleanup procedure: The following values are valid for the 'Command' parameter to start the cleanup procedure:

Value	Description
'CleanUp'	Removes all the temporary files and folders from the current MATLAB folder.
'CleanUpAll'	Removes not only the temporary files but also the generated real-time application. Only the Simulink model and the user files are kept.

#### Related topics

HowTos

- [How to Automate the Build Procedure](#) 

References

- *rtimp\_build* on page 128

## rti1xxx

#### Syntax

rti<XXXX>

#### Purpose

To activate a different RTI platform support and open the corresponding RTI block library.

#### Example


Type `rti1104` in the MATLAB Command Window to change to RTI1104 (DS1104 platform).

#### Related topics

HowTos

- [How to Activate a Specific Platform Support](#) 

## rtiver

<b>Syntax</b>	<code>rtiver</code>
<b>Purpose</b>	<p>To display the currently activated RTI platform support and the currently installed RTI version.</p> <div>  <ul style="list-style-type: none"> <li>■ You can use the <code>rti1xxx</code> command to activate a different RTI platform support. Refer to <i>rti1xxx</i> on page 126.</li> <li>■ You can use the <code>ver</code> command to display version information on The MathWorks products. For details, refer to the MATLAB Function Reference.</li> </ul> </div>
<b>Example</b>	<p>The following example shows a typical output of the <code>rtiver</code> command:</p> <pre>&gt;&gt; rtiver  RTI1104 5.3.6 (29-May-2006)</pre>

## rtimp\_blktargetcpunameget

<b>Syntax</b>	<pre>rtimp_blktargetcpunameget rtimp_blktargetcpunameget(&lt;mdlName&gt;, &lt;blockName&gt;)</pre>
<b>Purpose</b>	To return the name of the CPU which a given block belongs to.
<b>Result</b>	<p>If no arguments are specified, the name of the CPU is returned to which the currently selected block in the currently active model belongs.</p> <p>If <code>&lt;mdlName&gt;</code> and <code>&lt;blockName&gt;</code> are specified, the name of the CPU to which the block belongs is returned.</p>

<b>Description</b>	The function returns a struct. You will find the name of the CPU in the <code>targetcpu</code> field. If a block is assigned to multiple CPUs, the <code>targetcpu</code> field contains a list of CPU names. In addition, the <code>isvalidtargetcpu</code> field is 1 only if the block is assigned to a single CPU, otherwise it is zero.
<b>Valid for</b>	RTI-MP

## rtimp\_build

<b>Syntax</b>	<pre>[errorFlag, errorMsg] = rtimp_build mdlName, ['Command', &lt;commands&gt; [, &lt;Parameter&gt;, &lt;Value&gt; [, ...]]])</pre>
<b>Purpose</b>	To start the RTI-MP build procedure for a given application.
<b>Description</b>	<p>The command has the following outputs:</p> <p><b>errorFlag</b>    0 if no error occurred, otherwise a positive integer value.</p> <p><b>errorMsg</b>    Empty string if no error occurred, otherwise a string containing information about the error.</p> <p>The command has the following inputs:</p> <p><b>mdlName</b>    (String) The name of the model. The model to be built must be open.</p> <p><b>&lt;commands&gt;</b>    (String) The action to be carried out. Depending on this command an automated build, backup or cleanup procedure is carried out. Each of the possible commands has specific optional parameter-value pairs. See below for details on the commands and the related parameter-value pairs.</p> <p><b>&lt;Parameter&gt;, &lt;Value&gt;</b>    (String) Options for the given command, which are specified by parameter-value pairs. See below for details on the commands and the related parameter-value pairs.</p> <p>If you do not specify the 'Command', &lt;commands&gt; setting, <code>rtimp_build</code> defaults to <code>CodeGen&amp;Make</code> of the build procedure.</p>



- **Build procedure:** This is carried out in three phases: code generation, make and load. You can start each phase individually or combined with the others. The following values are valid for the 'Command' to start the build procedure:

Value	Description
'CodeGen' or 'C'	Generates the real-time code for the model.
'Make' or 'M'	Compiles and links the real-time code to the real-time application. CodeGen must be invoked beforehand.
'Load' or 'L'	Loads the real-time application to the hardware. Make must be invoked beforehand.
'CodeGen&Make' or 'CM'	Combines CodeGen and Make.
'Make&Load' or 'ML'	Combines Make and Load. CodeGen must be invoked beforehand.
'CodeGen&Make&Load' or 'CML'	Combines CodeGen, Make and Load.

The following parameter-value pairs are valid for the build procedure:

- **ApplNames** To specify the desired applications of the multiprocessor model:

Value	Description
'-all'	Processes all applications (default).
{ '<appl1>', '<appl2>', ... }	Processes only the specified applications.

- **KeepWorkingOnError** To select whether an error in one application stops the overall build procedure:

Value	Description
'on'	On an error occurring in one application, continue building the remaining applications (default).
'off'	On an error occurring, stop the overall build procedure.

The following parameter-value pairs are valid for the build procedure:

- **SkipTaskConfiguration** To select whether the Task Configuration dialog is opened interactively to let you confirm any changed tasks of the model, for example, newly added interrupt blocks:

Value	Description
'on'	Skips the interactive Task Configuration during the build process (default).
'off'	Opens the Task Configuration dialog if necessary and stops the build process.



When you add a new task to your model, the task automatically has a priority. However, this priority might not comply with your needs. Therefore, to avoid an unsuitable real-time application, you should check the task configuration of the model before you start the automated build procedure, especially if you select the `SkipTaskConfiguration`, 'on' setting.

- **LogOutput** To select whether the output from the MATLAB Command Window is saved to a text file:

Value	Description
'on'	Saves the output from the MATLAB Command Window to the <code>&lt;model&gt;_rtimp_log.txt</code> file.
'off'	Does not save the output from the MATLAB Command window (default).

- **MakeOptions** To specify additional options for the make process, which are appended to the existing make options. The value must be a cell array of strings, which lets you specify CPU-specific options:

Value	Description
{'',...}	No additional make options (default).
{ 'CC_OPTS',...}	Compiler options, for example, for two CPUs, { 'CC_OPTS="-H"', 'CC_OPTS="-H"' }
{ 'CC_OPT_OPTS',...}	Alternative compiler optimization, for example, for two CPUs, { 'CC_OPT_OPTS="-O1"', 'CC_OPT_OPTS="-O1"' }



You can also combine the 'CC\_OPTS' and 'CC\_OPT\_OPTS' settings in one value string. For example, for two CPUs,

```
{ 'CC_OPTS="-H" CC_OPT_OPTS="-O1"',  
  'CC_OPTS="-H" CC_OPT_OPTS="-O1"' }
```

- **Backup procedure:** The following value is valid for the 'Command' parameter to start the backup procedure:

Value	Description
'BackUp'	Creates a backup copy of the generated real-time application files that are necessary to load the application with ControlDesk.

The following parameter-value pairs are valid for the backup procedure:

- **BackUpFolder** To specify the location where the backup is saved. Make sure that the specified location exists before you start the backup procedure:

Value	Description
'<location>'	Saves the backup copy of the generated real-time application to this folder.

- **Cleanup procedure:** The following values are valid for the 'Command' parameter to start the cleanup procedure:

Value	Description
'CleanUp'	Removes all the temporary files and folders from the current MATLAB folder.
'CleanUpAll'	Removes not only the temporary files but also the generated real-time application. Only the Simulink model and the user files are kept.

The following parameter-value pairs are valid for the cleanup procedure:


The following parameter-value pairs are valid for the build procedure:

- **AppNames** To specify the desired applications of the multiprocessor model:

Value	Description
{ '-all' }	Processes all applications (default).
{ '<app1>', '<app2>', ... }	Processes only the specified applications.

## Related topics

### HowTos

- [How to Automate the Build Procedure](#) 

### References

- *rti\_build* on page 122

## set\_rti

<b>Syntax</b>	<pre>set_rti set_rti('-quiet') set_rti(&lt;ModelName&gt;) set_rti(&lt;ModelName&gt;,'-quiet')</pre>
<b>Purpose</b>	To configure the simulation parameters of a model for use with RTI.
<b>Result</b>	<p>This function configures the simulation parameters of the current model to settings suitable for code generation with the currently active platform, for example, the system target file, template makefile, make command, etc.</p> <p>If you do not want to reconfigure the current model, you can also specify a model by &lt;ModelName&gt;.</p> <p>This function makes use of modal dialogs which can be suppressed by using the optional '-quiet' flag.</p>
<b>Related topics</b>	<p>References</p> <ul style="list-style-type: none"> <li>• <i>rti_mdcleanup</i> on page 137</li> </ul>

## rti\_optionset

<b>Syntax</b>	[varValues]=rti_optionset(modelHandle,<parameters>,<values>)
<b>Purpose</b>	To set the values of options supported by the RTI options pages of the Real-Time Workshop dialog. For information, refer to <i>Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)</i> on page 27.

**Description**

The following table lists the supported parameters and their values:

Parameter Name	Possible Values
InitialSimState	RUN, PAUSE, STOP
ExecutionMode	real-time, time-scaled, as fast as possible
TimeScaleFactor	Any numeric value > 0
AssertionMode	OFF, WARN, STOP
TAStimulusEngineEnable	0, 1
EnableRealTimeTesting	0, 1
CCompilerCommonOpts	Any string
CCompilerOptimizationOptsPopup	Default, None, User-Defined
CCompilerOptimizationOpts	Any string <sup>1)</sup>
EnableDataSetStorage	0, 1 <sup>2)</sup>
LoadAppl	ON, OFF, FLASH <sup>3)</sup>
PlatformSelectionPopup	Auto, User-defined (bus), User-defined (network) <sup>4)</sup>
BoardName	Any string <sup>5)</sup>
NetworkClient	Any string <sup>6)</sup>
SlaveObject	Any string <sup>7)</sup>
TRCMaskParameters	0, 1
TRCGenerateLabels	0, 1
TRCGenerateVirtualBlocks	0, 1
TRCGenerateStates	0, 1
TRCGenerateDerivates	0, 1
TRCApplySubsystemPermissions	0, 1
TRCGenerateParamValues	0, 1

<sup>1)</sup> Only active if CCompilerOptimizationOptsPopup is set to User-defined.

<sup>2)</sup> Not supported for RTI1006, RTI1103, and RTI1104.

<sup>3)</sup> FLASH is not available for RTI1103.

<sup>4)</sup> User-defined (network) is not available for RTI1104 and RTI1401.

<sup>5)</sup> Only active if PlatformSelectionPopup is set to a value different from 'Auto'.

<sup>6)</sup> Only active if PlatformSelectionPopup is set to User-defined (network). This option is not supported for RTI1104 and RTI1401.

<sup>7)</sup> Not supported for RTI1005, RTI1006, RTI1104, and RTI1401.



The settings of the following parameter groups are interdependent:

- CCompilerOptimizationOptsPopup and CCompilerOptimizationOpts
- PlatformSelectionPopup, BoardName, and NetworkClient
- ExecutionMode and TimeScaleFactor

For example, the time-scaled simulation mode is only active, if the TimeScaleFactor parameter has a value > 0 **and** the ExecutionMode parameter is set to time-scaled.

#### Examples

```
rti_optionset(bdroot, 'TRCGenerateLabels', 1)
rti_optionset(bdroot, {'TRCGenerateLabels', 'LoadAppl'}, {1, 'OFF'})
```

## rti\_optionget

#### Syntax

```
[varValues]=rti_optionget(modelHandle, <parameters>)
```

#### Purpose

To get the values of options supported by the RTI options pages of the Real-Time Workshop dialog. For information, refer to *Real-Time Workshop Dialog (Model Configuration Parameters Dialogs)* on page 27.

**Description**

The following table lists the supported parameters and their values:

Parameter Name	Possible Values
InitialSimState	RUN, PAUSE, STOP
ExecutionMode	real-time, time-scaled, as fast as possible
TimeScaleFactor	Any numeric value > 0
AssertionMode	OFF, WARN, STOP
TAStimulusEngineEnable	0, 1
EnableRealTimeTesting	0, 1
CCompilerCommonOpts	Any string
CCompilerOptimizationOptsPopup	Default, None, User-Defined
CCompilerOptimizationOpts	Any string <sup>1)</sup>
EnableDataSetStorage	0, 1 <sup>2)</sup>
LoadAppl	ON, OFF, FLASH <sup>3)</sup>
PlatformSelectionPopup	Auto, User-defined (bus), User-defined (network) <sup>4)</sup>
BoardName	Any string <sup>5)</sup>
NetworkClient	Any string <sup>6)</sup>
SlaveObject	Any string <sup>7)</sup>
TRCMaskParameters	0, 1
TRCGenerateLabels	0, 1
TRCGenerateVirtualBlocks	0, 1
TRCGenerateStates	0, 1
TRCGenerateDerivates	0, 1
TRCApplySubsystemPermissions	0, 1
TRCGenerateParamValues	0, 1

<sup>1)</sup> Only active if CCompilerOptimizationOptsPopup is set to User-defined.

<sup>2)</sup> Not supported for RTI1006, RTI1103, and RTI1104.

<sup>3)</sup> FLASH is not available for RTI1103.

<sup>4)</sup> User-defined (network) is not available for RTI1104 and RTI1401.

<sup>5)</sup> Only active if PlatformSelectionPopup is set to a value different from 'Auto'.

<sup>6)</sup> Only active if PlatformSelectionPopup is set to User-defined (network). This option is not supported for RTI1104 and RTI1401.

<sup>7)</sup> Not supported for RTI1005, RTI1006, RTI1104, and RTI1401.





The settings of the following parameter groups are interdependent:

- CCompilerOptimizationOptsPopup and CCompilerOptimizationOpts
- PlatformSelectionPopup, BoardName, and NetworkClient
- ExecutionMode and TimeScaleFactor

For example, the time-scaled simulation mode is only active, if the TimeScaleFactor parameter has a value > 0 **and** the ExecutionMode parameter is set to time-scaled.

### Examples

```
optionValue=rti_optionget(bdroot,'TRCGenerateLabels')

optionValues=rti_optionget(bdroot,{ 'TRCGenerateLabels','LoadAppl' })
```

## rti\_mdldcleanup

### Syntax

```
rti_mdldcleanup
rti_mdldcleanup(<ModelName>)
```

### Purpose

To configure the simulation parameters of a model for use with a MATLAB/Simulink environment without RTI installation.

### Result

`rti_mdldcleanup` configures the simulation parameters of the current model (or the model specified by `<ModelName>`) to settings suitable for a MATLAB/Simulink environment without RTI installation.

### Description

When a model is configured for use with RTI, it contains not only RTI-specific blocks but also block-independent, RTI-specific settings. These remain in the model even if you remove all the RTI-specific blocks. If you open such a model with a non-RTI environment, MATLAB issues appropriate warnings. Therefore, you should use the `rti_mdldcleanup` function to remove all the RTI-specific settings from the model and avoid the warnings.

To use the `rti_mdldcleanup` function, the model must not contain any RTI blocks.

## Related topics

HowTos

- [How to Remove the RTI-Specific Settings from a Model](#)

References

- `set_rti` on page 133

## rti\_option TaskTransitionCheck

## Syntax

```
rti_option TaskTransitionCheck Relaxed
rti_option TaskTransitionCheck Strong
```

## Purpose

To enable or disable RTI's downward-compatible task transition check and Simulink's check for data transfer between an asynchronously triggered task and any arbitrary other task.

## Result

If set to `Relaxed`, Simulink does not check the data transfer between asynchronously triggered tasks (i.e., driven by an interrupt) and any other task. RTI accepts the old Uninterruptable Buffer block for modeling protected task transitions. The new task transition blocks cannot be used.

If set to `Strong` (default), the Simulink check is enabled, and the new task transition blocks have to be used for potentially interruptable signal lines.

## Valid for

RTI, RTI-MP

## Description

When the task transition check is `Strong`, Simulink issues an error message if it finds a direct connection between an interrupt block-driven task and another task. This ensures that you do not implement a non-buffered task transition by mistake.

For information on task transitions, see *Preparing a Function-Call Subsystem* in the *RTI and RTI-MP Implementation Guide*.

## Related topics

Basics

- [Preparing a Function-Call Subsystem](#)

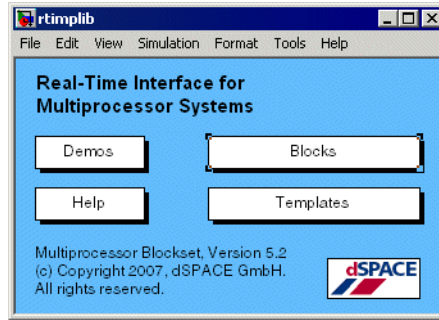
---

# General RTI and RTI-MP Libraries

	RTI and RTI-MP provide a number of general sublibraries, which contain blocks that are common to all dSPACE systems.
<b>RTI libraries</b>	The RTI library of the currently active RTI platform support is displayed when you enter <code>rti</code> in the MATLAB Command Window. Refer to Overview in the RTI Reference that corresponds to your hardware.
<b>Tasklib</b>	<b>The TaskLib sublibrary</b> offers RTI blocks for modeling interrupts and tasks in Simulink. It is accessible via the RTI libraries of the individual boards. Refer to <i>TaskLib Block Reference</i> on page 141.
<b>Extras</b>	<b>The Extras sublibrary</b> offers RTI blocks for special purposes, for example, the service code for ControlDesk. It is accessible via the RTI libraries of the individual boards. Refer to <i>Extras Block Reference</i> on page 164.

**rtimplib**

The RTI-MP library – **rtimplib** – for multiprocessor systems offers RTI-MP sublibraries with blocks for modeling multiprocessor models. It is displayed when you enter **rtimp**.



It provides RTI-MP blocks; for example, to mark communication connections between submodels. These RTI-MP blocks are designed to distribute the Simulink model to the different CPUs and can be copied to your Simulink model. The **rtimplib** also provides demo models, templates, and useful information.

The following components are available in the **rtimplib**:

**Demos** Contains example models. In some cases, you might want to use the demos as templates for your own model.

**Help** Displays this reference. The most important RTI-MP-related topics are described.

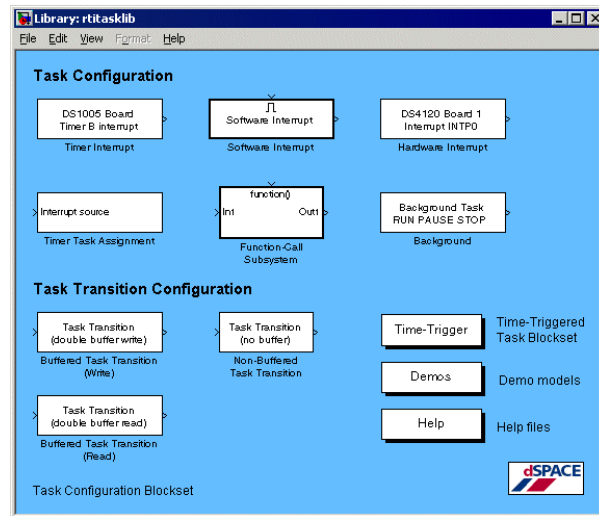
**Read Me** Displays last-minute information and important notes.

**Blocks** Contains all RTI-MP-specific blocks necessary to implement your multiprocessor model. Refer to *RTI-MP Block Reference* on page 172 for details.

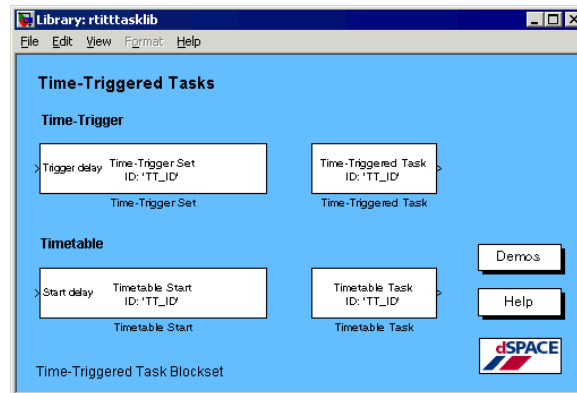
**Templates** Contains predefined template models for up to five CPUs so that every CPU has a direct connection to all other CPUs. You can modify their structures according to your needs, for example, by adding further IPC blocks or deleting unused IPC blocks.

# TaskLib Block Reference

After you double-click the TaskLib button in the RTI library, the rtitasklib library opens (an example is shown below).



The rtitasklib library contains the Time-Triggered Tasks blockset, which is shown below:

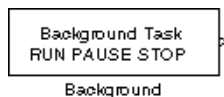


The Time-Triggered Tasks blockset is not available for RTI1104.

The TaskLib provides blocks for creating tasks driven by various interrupts.

Purpose	Refer to
Task configuration	
To create an interrupt-block-driven task or background task. To implement tasks driven by hardware interrupts or timer interrupts.  To implement a task driven by a signal of the simulation. To implement time-triggered tasks and timetable tasks that are started via a delay timer depending on another interrupt. These blocks are contained in the <b>Time-Triggered Task Blockset</b> sublibrary.  To synchronize the timer tasks to an external hardware interrupt. To execute a model part in the background.	<i>Function-Call Subsystem Block</i> on page 150 <ul style="list-style-type: none"> <li>• Interrupts and Slave DSP Interrupts in the <i>DS&lt;xxx&gt; RTI Reference</i> of your board</li> <li>• <i>Hardware Interrupt Block</i> on page 148</li> <li>• <i>Timer Interrupt Block</i> on page 153</li> </ul> <i>Software Interrupt Block</i> on page 152 <ul style="list-style-type: none"> <li>• <i>Time-Trigger Set Block</i> on page 156</li> <li>• <i>Time-Triggered Task Block</i> on page 158</li> <li>• <i>Timetable Start Block</i> on page 159</li> <li>• <i>Timetable Task Block</i> on page 160</li> </ul> <i>Timer Task Assignment Block</i> on page 161 <i>Background Block</i> on page 142
Task transition configuration	
To implement non-buffered data transfer. To implement a buffered data transfer	<i>Non-Buffered Task Transition Block</i> on page 151 <ul style="list-style-type: none"> <li>• <i>Buffered Task Transition (Read) Block</i> on page 143</li> <li>• <i>Buffered Task Transition (Write) Block</i> on page 146</li> </ul>

## Background Block



### Purpose

To execute a function-call subsystem in the background task of the real-time application.

### Description

Use this block to specify which parts of the Simulink model are executed in the background task asynchronously to the timer tasks of the model. You must specify at least one simulation state for which the function-call subsystem is executed: for the RUN, PAUSE or STOP simulation states, or any combination of the three.

### Dialog settings

**Execute subsystem for simState RUN** Select this checkbox if you want the function-call subsystem to be executed when the simulation state is set to RUN.

**Execute subsystem for simState PAUSE** Select this checkbox if you want the function-call subsystem to be executed when the simulation state is set to PAUSE.

**Execute subsystem for simState STOP** Select this checkbox if you want the function-call subsystem to be executed when the simulation state is set to STOP.



- Depending on the computational load of the foreground tasks and the fixed step size, the calculations performed in the background task can be either distributed over several base-sampling steps of the model or executed several times between two base-sampling steps.
- The service code for ControlDesk's virtual instruments is also executed in the background task (see `host_service`). If you make extensive use of Background blocks, this might slow down the updates of virtual instruments in ControlDesk layouts and the parameter download to the real-time processor.

## Related topics

### Basics

- [Assigning Parts of the Model to the Background Task](#)

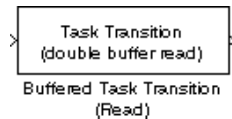
### HowTos

- [How to Assign Parts of the Model to the Background Task](#)

### References

- [Function-Call Subsystem Block](#) on page 150

## Buffered Task Transition (Read) Block



## Purpose

To implement a valid task transition for buffered data transfer between two tasks, of which at least one is an interrupt driven task.

**Description**

Since separate tasks can interrupt each other, the transfer of data vectors between them can also be interrupted, producing inconsistent data unless the data transfer is buffered.

With Simulink you can choose whether the transfer of data between different tasks is buffered to ensure data consistency or non-buffered to transfer the data as early as possible but at the cost of potentially inconsistent data vectors.


Together with the *Buffered Task Transition (Write) Block* on page 146, the Buffered Task Transition (Read) block allows you to implement a valid buffered task transition.

There are two possible applications for using the Buffered Task Transition (Read) block. In principle the Buffered Task Transition (Read) block mediates between a sending and a receiving block.

- The receiving block is placed inside a function-call subsystem. Then the Buffered Task Transition (Read) block is placed inside this function-call subsystem too, and directly connected to the Input block of the subsystem and the inport of the receiving block.



- The receiving block belongs to a timer task (i.e. only the sending block is placed inside a function-call subsystem). Then the Buffered Task Transition (Read) block is directly connected to the output of the sending function-call subsystem and the input of the receiving block.

For instructions on implementing a valid task transition, see *How to Prepare a Function-Call Subsystem* . For examples, see the demo models available from RTI's TaskLib.



- Task transition blocks support all data types except for complex ones.
- Task transition blocks do not support matrix signals. To transfer a matrix, use the Reshape block from the Simulink library, which converts the signal into a single vector. You can reshape it back into a matrix in the receiving task.



- Task transition blocks do not support frame-based signals, and there is no workaround available.
- Task transition blocks do not support buses containing signals with different data types. As a workaround, you can divide such busses into subbuses with equal data types.
- Task transition blocks do not support muxed signals with different sample times. As a workaround, you can divide such signals into subsignals with equal sample times.
- Task transition blocks inherit their data types from their sources.
- After simulation start, if the receiving task is executed before the sending task, the output value of the task transition block is not defined. To specify an initial value to be used in this case, you can use Simulink's IC (Initial Condition) block.




#### Dialog settings

**Sample time** Lets you specify the sample time for this block. Enter -1 for an inherited sample time.

## Related topics

## Basics

- [Introduction to Interrupt-Block-Driven Tasks and Task Transitions](#) 

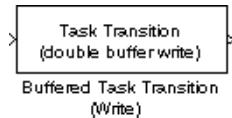
## HowTos

- [How to Prepare a Function-Call Subsystem](#) 

## References

- [Buffered Task Transition \(Write\) Block](#) on page 146
- [Function-Call Subsystem Block](#) on page 150
- [Non-Buffered Task Transition Block](#) on page 151

## Buffered Task Transition (Write) Block



## Purpose

To implement a valid task transition for buffered data transfer between function-call subsystems.

## Description

Since separate tasks can interrupt each other, the transfer of data vectors between them can also be interrupted, producing inconsistent data unless the data transfer is buffered.


With Simulink you can choose whether the transfer of data between different tasks is buffered to ensure data consistency or non-buffered to transfer the data as early as possible but at the cost of potentially inconsistent data vectors.

Together with the *Buffered Task Transition (Read) Block* on page 143, the Buffered Task Transition (Write) block allows you to implement a valid buffered task transition.

There are two possible applications for using the Buffered Task Transition (Write) block. In principle the Buffered Task Transition (Write) block mediates between a sending and a receiving block.

- The sending block is placed inside a function-call subsystem. Then the Buffered Task Transition (Write) block is placed inside this function-call subsystem too, and directly connected to the output of the sending block and the Output block of the subsystem.

- The sending block belongs to a timer task (i.e. only the receiving block is placed inside a function-call subsystem). Then the Buffered Task Transition (Write) block is directly connected to the output of the sending block and the inport of the receiving function-call subsystem.

For instructions on implementing a valid task transition, see *How to Prepare a Function-Call Subsystem* . For examples, see the demo models available from RTI's TaskLib.



- Task transition blocks support all data types except for complex ones.
- Task transition blocks do not support matrix signals. To transfer a matrix, use the Reshape block from the Simulink library, which converts the signal into a single vector. You can reshape it back into a matrix in the receiving task.



Reshape


- Task transition blocks do not support frame-based signals, and there is no workaround available.
- Task transition blocks do not support busses containing signals with different data types. As a workaround, you can divide such busses in subbusses with equal data types.
- Task transition blocks do not support muxed signals with different sample times. As a workaround, you can divide such signals in subsignals with equal sample times.
- Task transition blocks inherit their data types from their sources.

#### Dialog settings

**Sample time** Lets you specify the sample time for this block. Enter -1 for an inherited sample time.

## Related topics

## Basics

- [Introduction to Interrupt-Block-Driven Tasks and Task Transitions](#) 

## HowTos

- [How to Prepare a Function-Call Subsystem](#) 

## References

- [Buffered Task Transition \(Read\) Block](#) on page 143
- [Function-Call Subsystem Block](#) on page 150
- [Non-Buffered Task Transition Block](#) on page 151

## Hardware Interrupt Block



This block is only for use with the modular dSPACE systems and is intended for custom solutions only.

## Purpose

To implement a custom solution for an I/O board where no hardware interrupt block is available.

## Description

Several dSPACE I/O boards provide board-specific hardware interrupts. To make them available as trigger sources in a Simulink model, use the hardware interrupt blocks of the Controller board or I/O board you are addressing (see *Interrupts* and *Slave DSP Interrupts* in the *DS<xxxx> RTI Reference* that corresponds to your board).

For some I/O boards you might want to implement a custom solution that is not available via their RTI blocks. In that case, you can use the Hardware Interrupt block found in the TaskLib library of modular systems.

## Dialog settings

**Board type** Enter an I/O board in this edit field to access its hardware interrupt, for example, DS4120.

**Board number** 1 ... 16

**Interrupt number** INTP0 ... INTP7. This number must correspond to the interrupt number of the selected I/O board. You can find it in the interrupt table for the I/O board, see *DS<xxx> RTI Reference*.

**Perform sub-interrupt handling** Select this checkbox if sub-interrupt handling is required.

**Sub-interrupt number** Enter a positive integer value or 0.

This edit field is enabled only if the **Perform sub-interrupt handling** checkbox is selected.

**Use specific sub-interrupt handler** Select this checkbox if a board-specific sub-interrupt handler is needed.

This checkbox is enabled only if the **Perform sub-interrupt handling** checkbox is selected.

**Sub-interrupt handler function** Specify the name of the board-specific sub-interrupt handler function. This function must have a prototype that matches *dSPACE Real-Time Kernel's* definition for an interrupt handler.

This edit field is enabled only if the **Use specific sub-interrupt handler** checkbox is selected.

**Use custom code-generation file** Select this checkbox if additional code must be generated for the selected hardware interrupt block.

**TLC file name** Specify the name of the TLC file that is used to generate additional code for the selected interrupt block.





This edit field is enabled only if the **Use custom code-generation file** checkbox is selected.



- The TLC file you specify must be written according to the syntax of TLC files used for RTI's hardware interrupt blocks. For details, please contact <mailto:support@dSPACE.de>.
- Do not specify the extension of the TLC file.

---

**Related topics****HowTos**

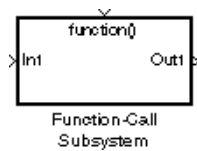
- [How to Combine Several Interrupt Sources](#) 
- [How to Implement Interprocessor Interrupts](#) 
- [How to Subschedule Tasks](#) 
- [How to Use Hardware Interrupts](#) 

**References**

- [Function-Call Subsystem Block](#) on page 150
- [Timer Task Assignment Block](#) on page 161
- [Timetable Start Block](#) on page 159
- [Time-Trigger Set Block](#) on page 156

---

## Function-Call Subsystem Block



---

**Purpose**

To insert a ready-to-use function-call subsystem triggered by a software or hardware interrupt.

---

**Dialog settings**

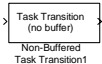
**Trigger type** The type of event that triggers the execution of the subsystem. Select “function-call”.

**Show output port** If this checkbox is selected, Simulink draws the Trigger block output and outputs the trigger signal.

Related topics

- Basics
- [Assigning Parts of the Model to the Background Task](#)
  - [Tasks Driven by Interrupt Blocks](#)
- HowTos
- [How to Combine Several Interrupt Sources](#)
  - [How to Implement Interprocessor Interrupts](#)
  - [How to Subschedule Tasks](#)
  - [How to Use Hardware Interrupts](#)
  - [How to Use Software Interrupts](#)
- References
- [Background Block](#) on page 142
  - [Buffered Task Transition \(Read\) Block](#) on page 143
  - [Buffered Task Transition \(Write\) Block](#) on page 146
  - [Hardware Interrupt Block](#) on page 148
  - [Non-Buffered Task Transition Block](#) on page 151
  - [Software Interrupt Block](#) on page 152
  - [Timer Interrupt Block](#) on page 153
  - [Timetable Task Block](#) on page 160
  - [Time-Triggered Task Block](#) on page 158

# Non-Buffered Task Transition Block




**Purpose** To implement a valid task transition for non-buffered data transfer between function-call subsystems.

**Description** Since separate tasks can interrupt each other, the transfer of data vectors between them can also be interrupted, producing inconsistent data unless the data transfer is buffered.

With Simulink you can choose whether the transfer of data between different tasks is buffered to ensure data consistency or non-buffered to transfer the data as early as possible but at the cost of potentially inconsistent data vectors.

The Non-Buffered Task Transition block allows you to implement a valid non-buffered task transition. It must be located on the same level as the receiving block or subsystem so that it is connected directly to the inport of the receiving block or subsystem.

For instructions on implementing a valid task transition, see *How to Prepare a Function-Call Subsystem* .




This block supports all data types and formats (vector, matrix, complex). The data types and formats of its input and output are identical.

#### Dialog settings

**Sample time** Lets you specify the sample time for this block. Enter -1 for an inherited sample time.

#### Related topics

Basics

- *Introduction to Interrupt-Block-Driven Tasks and Task Transitions* 

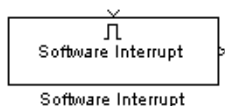
HowTos

- *How to Prepare a Function-Call Subsystem* 

References

- *Buffered Task Transition (Read) Block* on page 143
- *Buffered Task Transition (Write) Block* on page 146
- *Function-Call Subsystem Block* on page 150

## Software Interrupt Block



#### Purpose

To make software-generated interrupts available as trigger sources. Software interrupts can be used to build your own subscheduling of tasks.

#### Description

To start a Simulink subsystem by means of a software interrupt, connect the output of the Software Interrupt block to the trigger port of an function-call subsystem (see *Function-Call Subsystem Block* on page 150). The Software Interrupt block executes according to the settings of the Trigger and Enable ports. For example, if you use the default setting for the Software Interrupt block (Trigger port: “none”, Enable port: “yes”), the block triggers the corresponding function-call subsystem whenever the control signal has a positive value.



You can use any Simulink signal to trigger the execution of the Software Interrupt block, and thus to generate a software interrupt.

For further information on trigger and enable ports, refer to *Using Simulink* by The MathWorks.

Dialog settings

- Trigger port** Choose “none”, “falling”, “either”, or “function-call”. The default setting is “none”.
- Enable port** Choose “none” or “yes”. The default setting is “yes”.



Using the Software Interrupt block in the background task is not supported. That means you are not allowed to place the Software Interrupt block in a subsystem that is

- Connected to a Background Task block or
- Triggered from within a background task subsystem.

Related topics

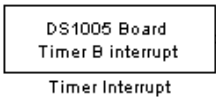
HowTos

- [How to Combine Several Interrupt Sources](#)
- [How to Implement Interprocessor Interrupts](#)
- [How to Subschedule Tasks](#)
- [How to Use Software Interrupts](#)

References

- [Function-Call Subsystem Block](#) on page 150
- [Timetable Start Block](#) on page 159
- [Time-Trigger Set Block](#) on page 156

# Timer Interrupt Block



Purpose

To make the timers of the platform available as interrupt sources.

As an example, the illustration above shows the Timer Interrupt block for the DS1005.

**Description**

Each platform provides various timers that are driven by the platform's bus clock. The Timer Interrupt block lets you use these timers as interrupt sources in a Simulink model. For example, if you specify "0.01" in the **Period** edit field of the block dialog, the selected timer generates one interrupt every 0.01 seconds.

The **Source** drop-down list of the Timer Interrupt block displays platform-independent names for the timer devices across all platforms. These names can differ from the names of the actual timer devices as shown in the table below. For example, if you specify "Timer C" of a DS1005, you actually select the board's Decrementer. The RTLib, feature and hardware references refer to the names of the timer devices.

The following table displays the bus clock frequencies that drive the corresponding timers. The bus clock frequency is referred to as BCLK.

Name in the Timer Interrupt Block	Hardware Device	Driving Frequency	Resolution	Count Range
<b>DS1005 Timing Devices (BCLK = 80 MHz)</b>				
Timer A interrupt	Timer A	BCLK/2	25 ns	32 bit
Timer B interrupt	Decrementer	BCLK/4	50 ns	32 bit
Timer C interrupt	Timer B	BCLK/4 <sup>1)</sup>	50 ns	32 bit
n/a	Time Base Counter	BCLK/4	50 ns	64 bit
<b>DS1005 Timing Devices (BCLK = 133.33 MHz)</b>				
Timer A interrupt	Timer A	BCLK/2	15 ns	32 bit
Timer B interrupt	Decrementer	BCLK/4	30 ns	32 bit
Timer C interrupt	Timer B	BCLK/4 <sup>1)</sup>	30 ns	32 bit
n/a	Time Base Counter	BCLK/4	30 ns	64 bit
<b>DS1006 Timing Devices (BCLK = 133.16 MHz)</b>				
Timer A interrupt	Timer A	BCLK/2	15.02 ns	32 bit
Timer B interrupt	Timer D	BCLK/4	30.04 ns	32 bit
Timer C interrupt	Timer B	BCLK/4	30.04 ns	32 bit
n/a	Time Stamp Counter	CPUCLK	0.455 ns	64 bit
<b>DS1103 Timing Devices (BCLK = 66.63 MHz)</b>				
Timer A interrupt	Decrementer	BCLK/4	60 ns	32 bit
Timer B interrupt	Timer A	BCLK/2	30 ns	32 bit
Timer C interrupt	Timer B	BCLK/4 <sup>2)</sup>	60 ns	32 bit
n/a	Time Base Counter	BCLK/4	60 ns	64 bit
<b>DS1103 Timing Devices (BCLK = 133.33 MHz)</b>				
Timer A interrupt	Decrementer	BCLK/4	30 ns	32 bit
Timer B interrupt	Timer A	BCLK/2	15 ns	32 bit
Timer C interrupt	Timer B	BCLK/4 <sup>2)</sup>	30 ns	32 bit
n/a	Time Base Counter	BCLK/4	30 ns	64 bit

Name in the Timer Interrupt Block	Hardware Device	Driving Frequency	Resolution	Count Range
<b>DS1104 Timing Devices (BCLK = 100 MHz)</b> (The bus clock of the DS1104 depends on the PCI bus clock of the Host PC.)				
Timer A interrupt	Timer 0	BCLK/8	80 ns	32 bit
Timer B interrupt	Timer 1	BCLK/8	80 ns	32 bit
n/a	Timer 2	BCLK/8	80 ns	32 bit
n/a	Timer 3	BCLK/8	80 ns	32 bit
Timer C interrupt	Decrementer	BCLK/4	40 ns	32 bit
n/a	Time Base Counter	BCLK/4	40 ns	64 bit
<b>DS1401 Timing Devices (BCLK = 50 MHz)</b>				
Timer A interrupt	Decrementer	BCLK/4	80 ns	32 bit
Timer B interrupt	Timer	BCLK/4 <sup>2)</sup>	80 ns	32 bit
n/a	Time Base Counter	BCLK/4	80 ns	64 bit
<b>DS1401 Timing Devices (BCLK = 100 MHz)</b>				
Timer A interrupt	Decrementer	BCLK/4	40 ns	32 bit
Timer B interrupt	Timer	BCLK/4 <sup>2)</sup>	40 ns	32 bit
n/a	Time Base Counter	BCLK/4	40 ns	64 bit


<sup>1)</sup> Timer C interrupt of the DS1005 is scalable from 1/512 to 1/4 BCLK. However, RTKernel (and therefore RTI) always uses the highest resolution at 1/4 BCLK.

<sup>2)</sup> Timer C interrupt of the DS1103 and Timer B interrupt of the DS1401 are scalable from 1/128 to 1/4 BCLK. However, RTKernel (and therefore RTI) always uses the highest resolution at 1/4 BCLK.

The minimum timer period is limited by the platform’s bus clock frequency. For example, if the bus clock frequency of the DS1005 is 80 MHz, the minimum timer period of Timer A is 25 ns.

The maximum timer period is limited by the platform’s bus clock and the resolution of the corresponding counter, which is 32 bits for all counters. For example, if the bus clock frequency of the DS1005 is 80 MHz, the maximum timer period of Timer A is 107.4 s (= 2^32 \* 25 ns).



You can find the value of the platform’s bus clock in ControlDesk’s *Properties: Platform Properties*  dialog.

Unit page

**Source** From this drop-down list, select a timer that serves as the interrupt source.

**Period** Enter the timer period in this edit field. The default period is 0.01 sec.



- Each timer displayed in the Source edit field can only be used once as an interrupt source in a Simulink model.
- If your model contains timer tasks, Timer A of your platform provides the necessary timer interrupts for the timer task(s). Therefore, you cannot use Timer A as an interrupt source for such a model. However, you can use Timer A if the timer tasks of your model are driven by another hardware interrupt via the Timer Task Assignment block (refer to *Timer Interrupt Block* on page 153).

## Related topics

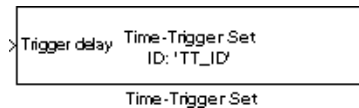
### HowTos

- [How to Combine Several Interrupt Sources](#)
- [How to Implement Interprocessor Interrupts](#)
- [How to Subschedule Tasks](#)
- [How to Use Hardware Interrupts](#)

### References

- *Function-Call Subsystem Block* on page 150
- *Properties: Platform Properties*
- *Timetable Start Block* on page 159
- *Time-Trigger Set Block* on page 156

## Time-Trigger Set Block



## Purpose

To set one or more task trigger delay times after which the corresponding time-triggered task is scheduled.

**Description**

When the Time-Trigger Set block is executed, it sets one or more task trigger delay times for the associated time-triggered task. The task trigger delay time (in seconds) is given by the Simulink signal connected to the inport of the Time-Trigger Set block. If a vector signal is connected, each vector element results in a separate task trigger delay time for the task.

When a task trigger delay time is reached, the time-triggered task is scheduled.

**Dialog settings**

**Time-Trigger ID** Associates the Time-Trigger Set block with its Time-Triggered Task block. Make sure to use a unique ID for each pair of Time-Trigger Set and Time-Triggered Task blocks.

**Sample time** Lets you specify a sample time. If the block is placed in a function-call subsystem, you must always use an inherited sample time (-1).

**Execute time-triggered task once at simulation start** If selected, the associated time-triggered task is executed once on simulation start independently of the execution of the Time-Trigger Set block.



This block is not available for RTI1104.

**Related topics**

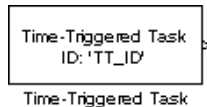
HowTos

- *How to Implement Time-Triggered Tasks* 

References

- *Hardware Interrupt Block* on page 148
- *Software Interrupt Block* on page 152
- *Timer Interrupt Block* on page 153
- *Time-Trigger Set Block* on page 156

## Time-Triggered Task Block



**Purpose** To execute the connected function-call subsystem whenever the task trigger delay time set by the corresponding Time-Trigger Set block is reached.

**Description** When the associated Time-Trigger Set block is executed, it sets one or more task trigger delay times for the time-triggered task. The task trigger delay time (in seconds) is given by the Simulink signal connected to the inport of the Time-Trigger Set block. If a vector signal is connected, each vector element results in a separate task trigger delay time for the task.

When a task trigger delay time is reached, the time-triggered task is scheduled.

**Dialog settings** **Time-Trigger ID** Associates the Time-Triggered task block with its Time-Trigger Set block. Make sure to use a unique ID for each pair of Time-Trigger Set and Time-Triggered Task blocks.



This block is not available for RTI1104.

### Related topics

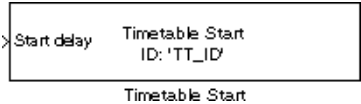
#### HowTos

- [How to Implement Time-Triggered Tasks](#)

#### References

- [Function-Call Subsystem Block](#) on page 150
- [Timer Task Assignment Block](#) on page 161
- [Time-Trigger Set Block](#) on page 156

## Timetable Start Block



Purpose	To specify the timetable start delay time and start the timetable consisting of the associated Timetable Task blocks.
Description	<p>When the Timetable Start block is executed, it sets the effective task trigger delay times for each associated timetable task. The effective task trigger delay times are the sum of:</p> <ul style="list-style-type: none"> <li>■ The timetable start delay time (in seconds) given by the Simulink signal that is connected to the Start delay inport of the Timetable Start block. If the Start delay inport is disabled, the timetable start delay time is 0.</li> <li>■ The individual task trigger delay times specified for the associated timetable tasks (see <i>Timetable Task Block</i> on page 160).</li> </ul>
Dialog settings	<p><b>Timetable ID</b> Associates the Timetable Start block with its Timetable Task blocks. Make sure to use a unique ID for each set of Timetable Start block and associated Timetable Task blocks.</p> <p><b>Sample time</b> Sample time Lets you specify a sample time. If the block is placed in a function-call subsystem, you must always use an inherited sample time (-1).</p> <p><b>Start timetable once at simulation start</b> If selected, the timetable is started once on simulation start independently of the execution of the Timetable Start block.</p> <p><b>Enable timetable start delay inport</b> If selected, the entire timetable is delayed by the value of the Simulink signal connected to the inport of the Timetable Start block.</p>



This block is not available for RTI1104.

## Related topics

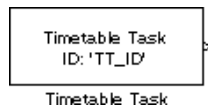
HowTos

- [How to Implement Timetables](#) 

References

- [Hardware Interrupt Block](#) on page 148
- [Software Interrupt Block](#) on page 152
- [Timer Interrupt Block](#) on page 153
- [Timestable Task Block](#) on page 160

## Timetable Task Block



## Purpose

To specify the task trigger delay time and execute the connected function-call subsystem when the task trigger delay time (plus the optional timetable start delay time of the Timetable Start block) is reached.

## Description

When the associated Timetable Start block is executed, it sets the effective task trigger delay times for each timetable task. The effective task trigger delay times are the sum of:


- The timetable start delay time (in seconds) given by the Simulink signal that is connected to the Start delay inport of the Timetable Start block. If the Start delay inport is disabled, the timetable start delay time is 0.
- The individual task trigger delay times specified for the timetable tasks. If a vector is specified as the Task trigger delays, one call of that timetable task is scheduled for each vector element.

## Dialog settings

**Timetable ID** Associates the Timetable Task block with its Timetable Start block. Make sure to use a unique ID for each set of Timetable Start block and associated Timetable Task blocks.




**Task trigger delays** Lets you specify one or more task trigger delay times for the timetable task (i.e., scalar or vector). The effective task trigger delay time is the sum of its task trigger delay and the timetable start delay time provided at the **Start delay** inport of the Timetable Start block.

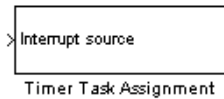


This block is not available for RTI1104.


Related topics

- HowTos
- *How to Implement Timetables* 
- References
- *Function-Call Subsystem Block* on page 150
  - *Timer Task Assignment Block* on page 161
  - *Timetable Start Block* on page 159

# Timer Task Assignment Block



**Purpose** To assign the connected hardware interrupt block, Time-Triggered Task block, or Timetable Task block as the trigger source for the model's timer task(s).



Do not connect a Timer Interrupt block or a Software Interrupt block to the Timer Task Assignment block.

**Description** Normally, a timer of your dSPACE real-time board provides the timer interrupts for your model's timer task(s). As an alternative, the Timer Task Assignment block allows you to specify external events as the interrupt source for the timer task(s). Therefore, you can easily synchronize the timer task execution with the events that occur on an I/O device connected to your dSPACE system, for example.

You can use the Timer Task Assignment block as follows:

- Connect it to any hardware interrupt block provided by the appropriate I/O library or by the RTI TaskLib. This synchronizes the timer tasks of the model to the interrupt signal of this hardware interrupt.
- Connect it to a Time-Triggered Task or Timetable Task block while the Time-Trigger Set block or Timetable Start block is driven by any hardware interrupt block. This also synchronizes the timer tasks of the model to the interrupt signal of the hardware interrupt block but using a variable delay time. During the real-time simulation you can easily modify the delay time.



The computations of the timer task are executed as normal using the time value of the simulation step size, as specified in the Solver dialog of the Model Explorer or in the Multiprocessor Setup dialog. However, the points in time when the computations are performed depend on the external trigger events. RTI cannot check whether the simulation step size of the model and the step size of the external trigger are equal. You must ensure identical step sizes yourself. Otherwise, the model is not calculated in real time.

---

#### Dialog settings

None

---

#### Related topics

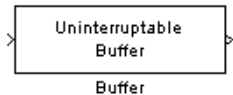
HowTos

- [How to Bind a Timer Task Explicitly to an Interrupt Block](#) 

References

- [Hardware Interrupt Block](#) on page 148
- [Timetable Task Block](#) on page 160
- [Time-Triggered Task Block](#) on page 158

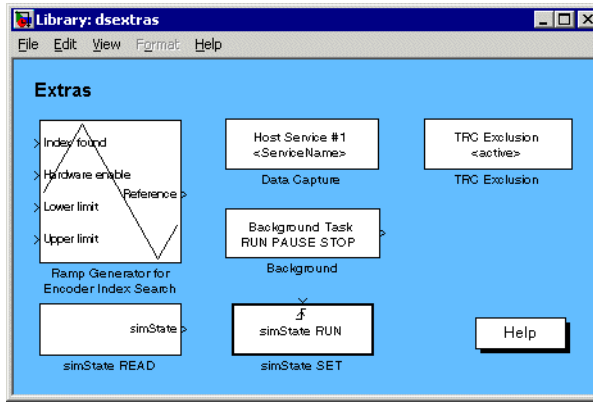
# Uninterruptable Buffer Block



Purpose	To ensure data consistency for the transfer of vectorized data between function-call subsystems.
Description	<p>This block is obsolete and only included for backwards compatibility. It is in future versions of RTI. Instead, use the Buffered Task Transition (Write) and Buffered Task Transition (Read) blocks. For details on valid task transitions, see <i>Preparing a Function-Call Subsystem</i> . For instructions on migrating a model to the new task transition blocks, refer to <i>Migrating to RTI 4.2 in dSPACE Release New Features and Migration</i> of Release 3.3.</p> <p>If you need to simulate a model that you cannot migrate to the new task transition blocks, follow the instructions given in <i>How to Enable Downward Compatibility for Task Transitions</i> .</p>
Dialog settings	None
Related topics	<p>Basics</p> <ul style="list-style-type: none"><li>• <i>Preparing a Function-Call Subsystem</i> </li></ul> <p>HowTos</p> <ul style="list-style-type: none"><li>• <i>How to Enable Downward Compatibility for Task Transitions</i> </li></ul> <p>References</p> <ul style="list-style-type: none"><li>• <i>Buffered Task Transition (Read) Block</i> on page 143</li><li>• <i>Buffered Task Transition (Write) Block</i> on page 146</li></ul>

## Extras Block Reference

After you double-click the **Extras** button in the RTI library, the **dsextras** library is displayed. The Extras library provides blocks that let you modify the data acquisition behavior or control the simulation state, for example.



The following table lists the blocks that are available from the Extras library.

Purpose	Refer to
Variable access	
To specify a host service for data capture.	<i>Data Capture Block</i> on page 165
To reduce the TRC file generation times and hide model parts.	<i>TRC Exclusion Block</i> on page 170
Simulation control	
To read the current simulation state.	<i>simState READ Block</i> on page 169
To set the current simulation state.	<i>simState SET Block</i> on page 169
Encoder index search	
To generate a ramp signal output for an encoder index search.	<i>Ramp Generator for Encoder Index Search Block</i> on page 167
Task configuration	
on page 141	<i>Background Block</i> on page 142

## Data Capture Block



### Purpose

To place the data acquisition host service code for ControlDesk in a specific task to control the rate at which data is sampled.

### Description

ControlDesk needs a special host service code to be executed in the real-time application. This code collects one set of data for each sampling step. By default, the service code for ControlDesk's data acquisition is executed in the fastest timer task.

If you use a Data Capture block, the default host service code for data sampling in the fastest timer task is omitted automatically.




- If your model is driven only by interrupt blocks, you need to place the Data Capture block in a triggered subsystem of your model to specify the task in which the host service code for the data acquisition is executed.
- Do not place a Data Capture block in the background task of a model or in a subsystem that is triggered by a Software Interrupt block placed in the background task. Otherwise, you get unpredictable results since the background task does not provide meaningful time stamps.

### Dialog settings

**Service number** The host service to be configured. The possible values are 1 ... 31.



If you use the RTI CAN MultiMessage Blockset with activated CAN monitoring option, make sure that the host service selected here differs from the host service (specified in the *RTICANMM GeneralSetup*  block) used for CAN monitoring with the CAN Navigator.

**Service name** Enter the service name in this edit field. This name is displayed on the **Capture** page of ControlDesk's CaptureSettings instrument.



If you run a multiprocessor system, the following applies:

- For distributed tracing the service name is used to define system-wide host services.
- If you want to perform distributed tracing for a specific interrupt block-driven task, you must place a Data Capture block in the corresponding triggered subsystem. Via IPC, IPI or Default CPU blocks, the data acquisition host service code generated for this Data Capture block is distributed to all CPUs that are involved in the task driven by the interrupt block. For distributed tracing, you must configure the IPC, IPI and Default CPU blocks so that the Data Capture block appears on all the CPUs of your system.

**Sample time** Enter the sample time at which the host service code is to be executed. Choose “-1” to inherit the sample time – this is the default setting – or any multiple of the “Fixed step size” chosen for the model. If you use the Data Capture block in triggered subsystems, select “-1” as the sample time.



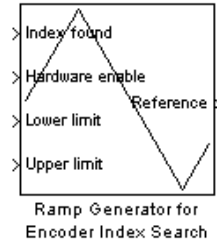
Do not specify sample time offsets.

## Related topics

HowTos

- [How to Modify Data Acquisition Behavior](#) 

## Ramp Generator for Encoder Index Search Block



This block has different behavior for the different RTIs.

### Purpose

To generate a ramp signal output that can be used as a reference signal during the index search of incremental encoders.



The MicroAutoBox does not provide an incremental encoder interface. Therefore, do not use this block when you create a Simulink model for your MicroAutoBox with RTI1401.

### I/O characteristics

- RTI supports data typing for this block. If the block's output is of Boolean type, the block can be used with all logical operators without using Data Type Conversion blocks. In this case, the block's input for the **Lower** and **Upper limit** – and for the modular systems also the Hardware Enable – have to be of Boolean type. The **Index found** input supports all data types.
- The data type of the outport always is double.

- The *ENC\_SW\_INDEX\_Cx* and *ENC\_SW\_INDEX\_Cx* blocks and also several *DS<xxx>ENC\_INDEX\_Bx\_Cy* blocks of the I/O boards for the modular systems provide the **Index found** output, which you can connect to the Index found input of the Ramp Generator block.

The Index found input of the Ramp Generator block has the following effect:

Index found input	Result
0	Ramp signal is generated
-1	Speed factor is applied
1	Ramp is switched off

If the **Hardware enable** input is set to 1, searching is enabled. If set to 0, searching is disabled.

For example, if the Type of index search of the DS1103ENC\_SW\_INDEX\_Cx or DS1104ENC\_SW\_INDEX\_Cx blocks is set to "Search index twice for speed-up" for RT1103/RT1104, and no index has been found yet – that is Index found is set to -1 – then the ramp generator **Reference output** can be used to speed up the movement of the controlled device. When the index has been found once – that is Index found is set to 0 – the Reference output can be used to slow down the movement of the controlled device to the value defined by the Reference output increment/decrement parameter.

- The **Lower Limit** and **Upper Limit** inputs are used to indicate whether a limit is reached (1) or not (0). If one of the limits is set to 1, the **Reference** output signal is inverted. For example, when the controlled device reaches the end of a linear course, its direction is changed.

#### Dialog settings

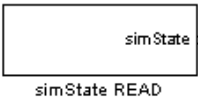
**Reference output increment/decrement** The value of the quantizing level of the ramp signal at the output.




**Speedup factor** Divided difference of the ramp signal: angle of lead  $\varphi = \arctan$  (Speedup factor).

**Sample time** The sample time of the task the ramp generator must be executed in. Values are "-1" (inherited) or any multiple of the "Fixed step size" chosen for the model.

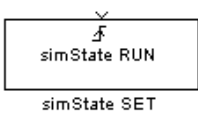


# simState READ Block



Purpose	To read the current simulation state in the Simulink model.
Description	The simState READ block reads the current simulation state. For further information on the simulation state, refer to <i>Simulation Control (RUN/STOP Mechanism)</i>  in the <i>RTI and RTI-MP Implementation Guide</i> .
Dialog settings	None
Related topics	<div>Basics</div> <ul style="list-style-type: none"><li>• <i>Simulation Control (RUN/STOP Mechanism)</i> </li></ul> <div>HowTos</div> <ul style="list-style-type: none"><li>• <i>How to Set and Read the Simulation State</i> </li></ul> <div>References</div> <ul style="list-style-type: none"><li>• <i>simState SET Block</i> on page 169</li></ul>

# simState SET Block




Purpose	To set the current simulation state from within the Simulink model.
---------	---

**Description**

This block is a conditionally executed subsystem. When the block is executed, the simulation state of the real-time application is set to the value specified in the block's dialog.

**RTI-MP**

This block is not valid for the slave CPUs of RTI-MP models.

For further information on the trigger and enable ports, refer to *Using Simulink* by The MathWorks. For further information on the simulation state, refer to *Simulation Control (RUN/STOP Mechanism)*  in the *RTI and RTI-MP Implementation Guide*.

**Dialog settings**

**Trigger port** Choose "none", "rising", "falling", "either", or "function-call".

**Enable port** Choose "none" or "yes".


**Set simState to** Choose "RUN", "PAUSE", or "STOP".

**Related topics**

Basics

- *Simulation Control (RUN/STOP Mechanism)* 

HowTos

- *How to Set and Read the Simulation State* 

References

- *simState READ Block* on page 169

## TRC Exclusion Block

TRC Exclusion  
<inactive>

TRC Exclusion

**Purpose**


To exclude all blocks (and other subsystems) in a subsystem from the generated variable description file (TRC file).

**Result**

All blocks in the subsystem where the TRC Exclusion block resides and all underlying subsystems are not generated into the TRC file.

**Description**

The block has two states: active and inactive, which you can set via the block dialog. The block itself displays the current state. You can add the block to any subsystem of a model.

When you build the model, TRC file generation stops inside the subsystem where the block resides. Only the outputs of the subsystem are available. If large parts of a model are excluded in this way, this reduces the size of the variable description file, which in turn results in faster loading in ControlDesk. You can also use this block to simply hide model parts from other users. For more information, refer to *Excluding Subsystems from the TRC File* .

Comments are added to the header of the generated TRC file, indicating whether the model contains active TRC Exclusion blocks. The affected subsystems are also listed.



Using the TRC Exclusion block in conjunction with the **Include mask and workspace parameters** option is not supported. For details, refer to *RTI variable description file options page* on page 34 (RTI) or *Variable Description File Options Page (CPU Options Dialog)* on page 61 (RTI-MP).

**Dialog settings**

**Block mode** Lets you set the state of the block. There are two states: active, which is set when 1 is entered; and inactive, which is set when 0 is entered. You can also enter the value as a workspace variable.

**Related topics**

## Basics

- *Excluding Subsystems from the TRC File* 

## References

- *Variable Description File (TRC File)* on page 109
- *Variable Description File Options Page (CPU Options Dialog)* on page 61

## RTI-MP Block Reference

RTI-MP provides a number of blocks for building multiprocessor models.

Purpose	Refer to
Simulation parameters	
To configure the various options relevant for the real-time simulation.	<i>Multiprocessor Setup Block</i> on page 180
Model distribution	
To assign model parts to the different CPUs and set up the interprocessor communication.	<i>Interprocessor Communication (IPC) Block</i> on page 174
To assign unconnected model parts to the different CPUs.	<i>Default CPU Block</i> on page 173
Task handling	
To transfer interrupts between CPUs.	<i>Interprocessor Interrupt (IPI) Block</i> on page 179

The following table summarizes which blocks are allowed on the different hierarchical levels of an RTI-MP model. The Multiprocessor Setup block is mandatory.

Hierarchical Level	Standard Blocks	Interrupt Blocks
Root level	Multiprocessor Setup block IPC block Default CPU block	Function-call subsystem Hardware interrupt block Software Interrupt block IPI block Background block
Level 1	Default CPU block	Function-call subsystem Hardware interrupt block Software Interrupt block Background block (local only)
Levels 2+	(None)	Function-call subsystem Hardware interrupt block Software Interrupt block Background block (local only)
Function-call subsystem	IPC block Default CPU block	Function-call subsystem Hardware interrupt block Software Interrupt block
Level 1 below function-call subsystem	Default CPU block	Function-call subsystem Hardware interrupt block Software Interrupt block

Hierarchical Level	Standard Blocks	Interrupt Blocks
Levels 2+ below function-call subsystem	(None)	Function-call subsystem Hardware interrupt block Software Interrupt block



Background blocks that are placed below root level can be used only locally. This means that the corresponding function-call subsystem should not contain any RTI-MP blocks.

## Default CPU Block

Default CPU:

**Purpose** To assign CPU identities to unconnected blocks on the top level of the Simulink model or within triggered subsystems.

**Description** Each Simulink block of a multiprocessor system inherits its CPU identity from a connected IPC block. The Default CPU block is used to assign the CPU identity to unconnected blocks in a multiprocessor system. Therefore, a Default CPU block can be located on the model root to assign all unconnected blocks to the specified CPU, and in any subsystem (the second hierarchical level of a model) to assign only the current subsystem to a specific CPU.

Since triggered subsystems can be computed on more than one CPU, Default CPU blocks can also be used in triggered subsystems. However, it is not possible to have more than one Default CPU block on the model root or on the top level of a (triggered) subsystem.



If a multiprocessor model contains unconnected blocks but not a Default CPU block, RTI-MP issues a warning but no error, and automatically assigns them to the master CPU.


**Adding a CPU** RTI-MP determines the names of the CPUs in a model via the CPU names specified in IPC blocks, IPI blocks and Default CPU blocks. Each CPU name found in such a block is listed in the Multiprocessor Setup dialog and gets its own CPU's page to set up the application-specific properties, such as Name and Step size multiple. To add a new CPU to the system, it is sufficient to specify its name in an IPC block, IPI block or Default CPU block.

**Dialog settings**

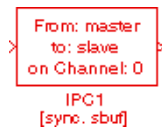
**Default CPU** Specify the name of the default CPU.

**Related topics**

HowTos

- *How to Define a Default CPU* 

## Interprocessor Communication (IPC) Block

**Purpose**

To define the parameters of a communication channel between the CPUs of a multiprocessor system.



Before the parameters of the corresponding channel and the IPC blocks belonging to the current connection can be changed, the **Multiprocessor Setup** dialog must be closed.


**Description**

IPC blocks must be located on the top level of the model or in triggered subsystems. The source and destination CPU plus the channel number are displayed on the IPC block. IPC blocks can be copied by drag & drop like any other Simulink block, to get more connector blocks for one communication channel or establish new communication channels and new CPUs. All the Simulink blocks of a multiprocessor system inherit their CPU identities from a connected IPC block.

To delete an IPC connection, just delete all the IPC blocks of the connection.

**Simulink simulation** Each IPC block contains an S-function that checks the data types of the input signals. This S-function behaves exactly like the standard Simulink Mux and Demux blocks, except in buffering or unbuffering mode. In buffering or unbuffering mode, it emulates the buffering and unbuffering of signals. Therefore, Simulink simulation can always be performed with the same results as a multiprocessor real-time simulation.

**Adding a CPU** RTI-MP determines the names of the CPUs in a model via the CPU names specified in IPC blocks, IPI blocks and Default CPU blocks. Each CPU name found in such a block is listed in the Multiprocessor Setup dialog and gets its own CPU's page to set up the application-specific properties, such as Name and Step size multiple. To add a new CPU to the system, it is sufficient to specify its name in an IPC block, IPI block or Default CPU block.

For details on communication (IPC) connections, see *Interprocessor Communication*  in the *RTI and RTI-MP Implementation Guide*.

#### Data buffering and unbuffering


You can also use the IPC block to achieve data buffering or unbuffering. The buffering or unbuffering mode is selected automatically for an IPC connection as follows:

- Step size of source CPU < step size of destination CPU  
The IPC connection is operated in buffering mode. For example, if the step size multiples are 1 for the source CPU and 10 for the destination CPU, and the number of buffered samples is 10, a vector of 10 input values is collected and passed to the destination CPU at every 10<sup>th</sup> simulation step.
- Step size of source CPU > step size of destination CPU  
The IPC connection is operated in unbuffering mode. For example, if the step size multiples are 10 for the source CPU and 1 for the destination CPU, and the number of buffered samples is 10, a vector of 10 values is passed from the source to the destination CPU at every 10<sup>th</sup> simulation step. In each basic simulation step, the destination CPU uses one of the ten values as the model input.

You can use the **Buffered samples** edit field of the Communication Channel Setup dialog to define how many signal samples are to be buffered or unbuffered on an IPC connection.



Make sure that you transfer signals of the double data type and use the swinging buffer protocol. The IPC block does not support other data types or the virtual shared memory protocol if operated in the buffering or unbuffering mode.

See *Example of buffering/unbuffering via an IPC block*  in the *RTI and RTI-MP Implementation Guide* for an instructive example of data buffering/unbuffering.

#### Communication channel frame

**Connection: Source CPU** Lets you specify the name of the source CPU (origin of the **IPC block**). CPU names are restricted to eight alphanumeric characters.

To add another CPU to the multiprocessor model, simply enter a new CPU name. The parameters of the new CPU, such as application name and solver, can be entered the next time the **Multiprocessor Setup** dialog is opened.

**Connection: Destination CPU** Lets you specify the name of the target CPU. CPU names are restricted to eight alphanumeric characters.



To add another CPU to the multiprocessor model, simply enter a new CPU name. The parameters of the new CPU, such as application name and solver, can be entered the next time the Multiprocessor Setup dialog is opened.

**Connection: Channel** Lets you specify the channel number.


You can select any non-negative integer number as the channel number.

**Step size multiple** Specify the step sizes of the communication channel as integer multiples of the basic real-time step size. The step sizes have to be a multiple of the step size of the CPU. The **Step size multiple** must be specified independently for the **Source** and the **Destination CPU**.

**Protocol** From the drop-down list, choose one of the following protocols:


- *Swinging Buffer Protocol*  (synchronized or unsynchronized)
- *Virtual Shared Memory Protocol* 



For details on the protocols, see *Interprocessor Communication*  in the *RTI and RTI-MP Implementation Guide*.

**Number of signals** The number of signals to be transferred via a communication channel is displayed. It results from the values specified in the **Width** fields of the **Member IPC blocks** belonging to the current communication channel.

**Buffered samples** Specify the number of buffered signal samples to be sent per cycle.

Normally, the number of buffered samples is equal to 1, which means no buffering/unbuffering is performed. If the number of buffered samples is greater than 1, the connection operates in buffering or unbuffering mode, depending on the **Step size multiple** of the **Source CPU** and **Destination CPU** of the current connection (see *Communication with a Sample Time Transition*  for details).



The communication protocol must be set to **Synchronized Swinging Buffer** or **Unsynchronized Swinging Buffer** if you want to operate an IPC block in buffering/unbuffering mode. The entry for the Data type for this IPC connection must be "double."

#### Member IPC blocks frame

**Show Block** Highlights the current IPC block in the model.

**Name** The Simulink names of all IPC blocks belonging to the current connection are displayed. The IPC block that is currently highlighted in the model, is displayed in bold letters.

**Width** Enter the signal width of the corresponding IPC block.

The **Number of Signals** of the current communication channel is automatically adjusted to the entries in the **Width** windows of the IPC blocks.

If a vector of  $N$  signals is to be transmitted by means of an IPC block, its width must be given in brackets as " $[n_1 \ n_2 \ \dots \ n_N]$ ", where  $n_1 \ \dots \ n_N$  represent the widths of the signal vectors of the  $n^{\text{th}}$  IPC block. The total number of signals to be sent over the corresponding block is the sum of all elements of the width vector. For example, if two signal vectors with 3 and 4 elements are to be transmitted, the signal width is "[3 4]". A specification of "[3]" corresponds to a single signal vector of 3 elements, whereas "3" (without brackets) represents 3 scalar signals. Square brackets always indicate vector signals. The number of inports and outports of the IPC block is adjusted according to the number of signals entered as **Width**.

**Data type** All MATLAB-supported data types can be selected for the corresponding IPC block by specifying one of the following data types from the drop-down list: "double," "single," "int8," "uint8," "int16," "uint16," "int32," "uint32," and "Boolean". The default data type is "double".



- If you want an IPC connection to operate in buffering or unbuffering mode, its data type must be "double". The **Protocol** must be set to Synchronized or Unsynchronized Swinging Buffer.
- If the data type of an IPC block and the input signal(s) do not match, an error message is sent by Simulink. To avoid this, either adapt the data type of the IPC block to the input or provide an input signal of the desired data type.

**Change connection** Opens the Change connection dialog (see *Change Connection Dialog* on page 44).

## Related topics

### Basics

- [Interprocessor Communication](#)
- [Swinging Buffer Protocol](#)
- [Virtual Shared Memory Protocol](#)

### HowTos


- [How to Implement Interprocessor Communication](#)

### References

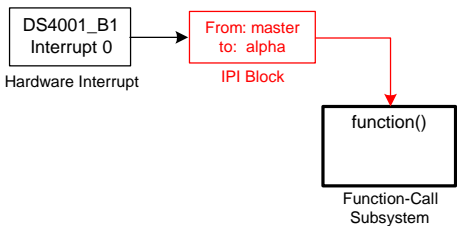
- [Change Connection Dialog](#) on page 44

# Interprocessor Interrupt (IPI) Block



Purpose	To specify the interrupt source CPU and the target CPUs in multiprocessor systems.
	<div><div>IPI blocks must be located on the top level of the RTI-MP model.  For local interrupts, i.e., if the source CPU and the destination CPU are identical, no IPI block is required.</div></div>

Description	In multiprocessor systems hardware and software interrupts can trigger tasks on more than one CPU. Therefore, Hardware and Software interrupt blocks from the specific RTI board library can only be used together with IPI blocks. These IPI blocks must be placed between the Hardware or Software interrupt block and the triggered subsystem. See the illustration below.
-------------	---



**Adding a CPU** RTI-MP determines the names of the CPUs in a model via the CPU names specified in IPC blocks, IPI blocks and Default CPU blocks. Each CPU name found in such a block is listed in the Multiprocessor Setup dialog and gets its own CPU's page to set up the application-specific properties, such as Name and Step size multiple. To add a new CPU to the system, it is sufficient to specify its name in an IPC block, IPI block or Default CPU block.

Dialog settings	<div><b>Maximum number of Target CPUs</b> Specify the number of involved target CPUs.</div> <div><b>Source CPU</b> Specify the source CPU of the interprocessor interrupt.</div>
-----------------	--

**Target CPU 1 ... Target CPU x** Specify the target CPU(s) of the interprocessor interrupt.

#### Related topics

HowTos

- *How to Implement Interprocessor Interrupts* 

## Multiprocessor Setup Block

**Multiprocessor Setup**

The **Multiprocessor Setup** dialog is displayed by double-clicking the **Multiprocessor Setup** block (see *Multiprocessor Setup Dialog* on page 45). This dialog lets you:

- Set the global parameters of the multiprocessor Simulink model
- Set the CPU-specific parameters of the multiprocessor Simulink model
- Change the target multiprocessor system
- Get further information about your model and add comments




This block must be on the top level of every RTI-MP model.



You can double-click the **Multiprocessor Setup** block found in the RTI-MP **Blocks** library to create a new RTI-MP model from scratch: A new Simulink model containing a Multiprocessor Setup block and two IPC blocks is opened. You can insert your own Simulink blocks and add, delete and change communication channels and CPUs.

#### Related topics

HowTos

- *How to Configure an RTI-MP Model* 

References

- *Multiprocessor Setup Dialog* on page 45

**A**

Advanced page  
 Multiprocessor Setup dialog 54  
 assertion mode 87

**B**

Background block 142  
 blocks  
   background 142  
   buffered task transition  
     read 143  
     write 146  
   data capture 165  
   default CPU 173  
   function-call subsystem 150  
   hardware interrupt 148  
   interprocessor  
     communication 174  
     interrupt 179  
   model verification 24, 74  
   multiprocessor setup 180  
   non-buffered task transition 151  
   ramp generator for encoder index  
     search 167  
     simState  
       read 169  
       set 169  
   software interrupt 152  
   timer interrupt 153  
   timer task assignment 161  
   timetable start 159  
   time-trigger set 156  
   time-triggered task 158  
   TRC exclusion 170  
   uninterruptable buffer 163  
 blocks for RTI-MP 172  
 blocks timetable task 160  
 buffered samples 177  
 Buffered Task Transition (Read) block 143  
 Buffered Task Transition (Write) block 146  
 buffering 175  
 build.bat file 96

**C**

category  
   rti general build options 32  
   rti load options 33  
   rti variable description file options 34  
 Change Connection dialog 44  
 clean.bat file 96  
 Communication Channels display 44  
 CPU identity 173

CPU's page 49  
 currentTime 82

**D**

data buffering 175  
 Data Capture block 165  
 data set storage 33  
 data type specifications 118  
 Default CPU block 173  
 -DFIRST\_SIMSTEP\_INCREASEMENT 58  
 Diagnostics dialog  
   RTI 23  
   RTI-MP 73  
 Documentation page 56  
 download.bat file 97  
 -DRTIMP\_GL\_NO\_TIMEOUT 58  
 DSFINISH.M file 103  
 DSSTARTUP.M file 102

**E**

encoder index search 167  
 errorNumber 83  
 External Mode Control Panel  
   RTI 40  
   RTI-MP 77  
 Extras 164

**F**

file overview 94  
 files  
   build.bat 96  
   clean.bat 96  
   download.bat 97  
   DSFINISH.M 103  
   DSSTARTUP.M 102  
   IPC.C 95  
   linker command 108  
   LK 108  
   MAP 98  
   PPC 97  
   RTI.C 95  
   RTI.MK 96  
   RTI.PRJ 96  
   SDF 97  
   SIMENG.C 95  
   STARTUP.M 101  
   TH.C 94  
   TRC 109  
   user makefile 105  
   user system description file 108  
   user variable description file 118  
   User-Code 103

*files*

  USR.C 103  
   USR.MK 105  
   USR.SDF 108  
   USR.TRC 118  
   variable description file 109  
   X86 97  
 finalTime 83  
 FIRST\_SIMSTEP\_INCREASEMENT 58

**H**

Hardware Implementation dialog  
   RTI 25  
 Hardware Interrupt block 148

**I**

interrupt  
   function 150  
   Hardware Interrupt block 148  
   Software Interrupt block 152  
 IPC block 174  
 IPC.C file 95  
 IPI block 179

**L**

library 139  
   EXTRAS 164  
   RTI 172  
   TaskLib 141  
 linker command file 108  
 LK file 108

**M**

Main page 46  
 MAP file 98  
 Model Configuration Parameters dialogs  
   RTI 17, 68  
   RTI-MP 67  
 Model Parameters Configuration dialog  
   RTI 37  
   RTI-MP 76  
 Model Verification blocks 24, 74  
 modelStepSize 84  
 multiprocessor  
   setup block 180  
   setup dialog 45  
   topology setup dialog 65

**N**

name of member IPC block 177  
 Non-Buffered Task Transition block 151

number of signals 177

## O

Optimization dialog  
     RTI 21  
     RTI-MP 69  
 overrunCheckType 84  
 overrunCount 85

## P

permissions  
     subsystem 36, 64  
 PPC file 97  
 priority 86  
 protocol for RTI-MP 176

## R

Ramp Generator for Encoder Index Search  
 block 167  
 Real 75  
 Real-Time Workshop dialog  
     RTI 27  
 Rename CPU dialog 64  
 RTI 172  
 rti general build options category 32  
 rti load options category 33  
 RTI Task Configuration dialog  
     RTI 38  
     RTI-MP 56  
 rti variable description file options  
 category 34  
 RTI.C file 95  
 RTI.MK file 96  
 RTI.PRJ file 96  
 rti\_build 122  
 rti\_option  
     TaskTransitionCheck 138  
 rtiAssertionMode 87  
 rtilib  
     EXTRAS 164  
     TaskLib 141  
 rtmp\_build 128  
 RTIMP\_GL\_NO\_TIMEOUT 58

## S

sampleTime 88  
 SDF file 97  
 Signal Properties dialog  
     RTI 42  
     RTI-MP 78  
 signal width of an IPC block 177  
 SIMENG.C file 95

simState 88  
     simState READ block 169  
     simState SET block 169  
 simulation state 88  
     simState READ block 169  
     simState SET block 169  
 Software Interrupt block 152  
 STARTUP.M file 101  
 state 89  
     simState 88  
     task 89  
 SubArray 115  
 subscheduling of tasks 152  
 subsystem permissions 36, 64

## T

task  
     state 89  
 Task Configuration dialog  
     RTI 38  
     RTI-MP 56  
 task overrun 39, 57, 58, 84  
 task transition  
     Buffered Task Transition (Read)  
     block 143  
     Buffered Task Transition (Write)  
     block 146  
     Non-Buffered Task Transition block 151  
 TaskLib 141  
 TH.C file 94  
 Timer Interrupt block 153  
 Timer Task Assignment block 161  
 Timetable Start block 159  
 Timetable Task block 160  
 Time-Trigger Set block 156  
 Time-Triggered Task block 158  
 TRC Exclusion block 170  
 TRC file 109  
 tunable parameters  
     dialog  
         RTI 37  
         RTI-MP 76  
 turnaroundTime 90

## U

unbuffering 175  
 Uninterruptable Buffer block 163  
 user makefile 105  
 user system description file 108  
 user variable description file 118  
 User-Code file 103  
 USR.C file 103  
 USR.MK file 105  
 USR.SDF file 108

USR.TRC file 118  
 usr\_background 104  
 usr\_initialize 104  
 usr\_input 104  
 usr\_output 104  
 usr\_sample\_input 104  
 usr\_terminate 104

## V

variable description file 109  
 variable description file groups 113  
 variables 81

## X

X86 file 97