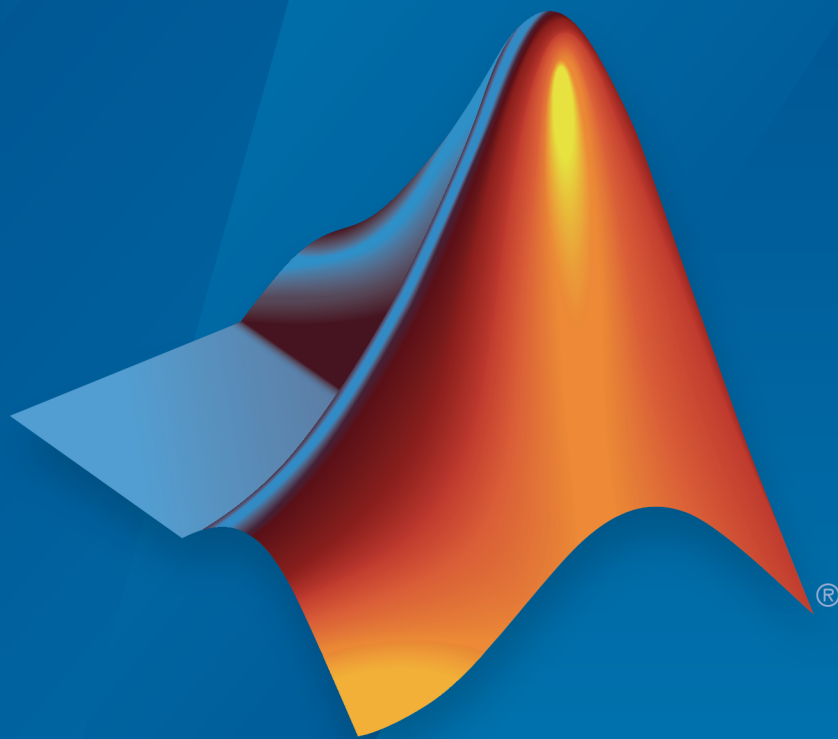


Simulink<sup>®</sup> Real-Time<sup>™</sup>

Reference



MATLAB<sup>®</sup>&SIMULINK<sup>®</sup>

R2016a



## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



The MathWorks, Inc.  
3 Apple Hill Drive  
Natick, MA 01760-2098

*Simulink<sup>®</sup> Real-Time<sup>™</sup> Reference*

© COPYRIGHT 2002–2016 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

**FEDERAL ACQUISITION:** This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### **Trademarks**

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### **Patents**

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

## Revision History

March 2007	Online only	New for Version 3.2 (Release 2007a)
September 2007	Online only	Updated for Version 3.3 (Release 2007b)
March 2008	Online only	Updated for Version 3.4 (Release 2008a)
October 2008	Online only	Updated for Version 4.0 (Release 2008b)
March 2009	Online only	Updated for Version 4.1 (Release 2009a)
September 2009	Online only	Updated for Version 4.2 (Release 2009b)
March 2010	Online only	Updated for Version 4.3 (Release 2010a)
April 2011	Online only	Updated for Version 5.0 (Release 2011a)
September 2011	Online only	Updated for Version 5.1 (Release 2011b)
March 2012	Online only	Revised for Version 5.2 (Release 2012a)
September 2012	Online only	Revised for Version 5.3 (Release 2012b)
March 2013	Online only	Revised for Version 5.4 (Release 2013a)
September 2013	Online only	Revised for Version 5.5 (Release 2013b)
March 2014	Online only	Revised for Version 6.0 (Release 2014a)
October 2014	Online only	Revised for Version 6.1 (Release 2014b)
March 2015	Online only	Revised for Version 6.2 (Release 2015a)
September 2015	Online only	Revised for Version 6.3 (Release 2015b)
March 2016	Online only	Revised for Version 6.4 (Release 2016a)



## Configuration Parameters

1

<b>Code Generation Configuration Parameters</b> .....	<b>1-2</b>
Simulink Real-Time Options Pane .....	1-3
Build for default target computer .....	1-6
Specify target computer name .....	1-7
Automatically download application after building .....	1-8
Name of Simulink Real-Time object created by build process ..	1-8
Use default communication timeout .....	1-10
Specify the communication timeout in seconds .....	1-11
Execution mode .....	1-12
Real-time interrupt source .....	1-13
I/O board generating the interrupt .....	1-14
PCI slot (-1: autosearch) or ISA base address .....	1-18
Log Task Execution Time .....	1-19
Signal logging data buffer size in doubles .....	1-20
Number of profiling events (each uses 20 bytes) .....	1-22
Double buffer parameter changes .....	1-23
Load a parameter set from a file on the designated target file system .....	1-24
File name .....	1-25
Generate INCA/CANape extensions .....	1-26
Enable Stateflow animation .....	1-27

**2**

**Simulink Real-Time Explorer Instrumentation**

**3**

<b>Instrumentation for Real-Time Applications</b> .....	<b>3-2</b>
Instrument Selection and Binding .....	3-4
Layout Elements .....	3-7
<b>Explorer Configuration Exported to Run Outside MATLAB</b>	<b>3-10</b>
<b>Guidelines for Exporting Explorer Configuration</b> .....	<b>3-12</b>
Execution Environment .....	3-12
Signal Groups .....	3-13
Parameter Groups .....	3-13
Instrument Panels .....	3-13
Window Layout .....	3-13
<b>Create Instrument Panel</b> .....	<b>3-14</b>
<b>Configure Instrument for Parameter Tuning</b> .....	<b>3-15</b>
<b>Configure Instruments for Signal Display</b> .....	<b>3-18</b>
<b>Save and Load Environment Properties</b> .....	<b>3-21</b>
<b>Save and Load Instrument Panels</b> .....	<b>3-22</b>
<b>Save and Restore Layouts</b> .....	<b>3-23</b>
<b>Run Instrumented Model</b> .....	<b>3-24</b>
<b>Prepare Explorer Environment for Export</b> .....	<b>3-27</b>
<b>Prepare Instrument Panel Configuration for Export</b> .....	<b>3-29</b>
<b>Export Explorer Configuration</b> .....	<b>3-31</b>

## Simulink Real-Time Explorer Instruments

4

## Target Computer Command-Line Interface Reference

5

<b>Target Computer Commands</b> .....	5-2
Target Object Function Commands .....	5-2
Target Object Property Commands .....	5-3
Scope and Video Object Function Commands .....	5-4
Scope Object Property Commands .....	5-6
Aliasing with Variable Commands .....	5-10

## Simulink Real-Time Performance Advisor Checks

6

<b>Simulink Real-Time Performance Advisor Checks</b> .....	6-2
Performance Advisor Check Overview .....	6-2
Baseline .....	6-2
System Target File Compatibility .....	6-3
Profiling Settings .....	6-3
Check Target .....	6-3
Real-Time Performance Baseline .....	6-4
Determine minimum sample time .....	6-4
Real-Time .....	6-5
Outport Logging .....	6-5
EtherCAT Synchronous SDO .....	6-5
Concurrent execution .....	6-6
Final Validation .....	6-6





# Configuration Parameters

---

## Code Generation Configuration Parameters

### In this section...

- “Simulink Real-Time Options Pane” on page 1-3
- “Build for default target computer” on page 1-6
- “Specify target computer name” on page 1-7
- “Automatically download application after building” on page 1-8
- “Name of Simulink Real-Time object created by build process” on page 1-8
- “Use default communication timeout” on page 1-10
- “Specify the communication timeout in seconds” on page 1-11
- “Execution mode” on page 1-12
- “Real-time interrupt source” on page 1-13
- “I/O board generating the interrupt” on page 1-14
- “PCI slot (-1: autosearch) or ISA base address” on page 1-18
- “Log Task Execution Time” on page 1-19
- “Signal logging data buffer size in doubles” on page 1-20
- “Number of profiling events (each uses 20 bytes)” on page 1-22
- “Double buffer parameter changes” on page 1-23
- “Load a parameter set from a file on the designated target file system” on page 1-24
- “File name” on page 1-25
- “Generate INCA/CANape extensions” on page 1-26
- “Enable Stateflow animation” on page 1-27

## Simulink Real-Time Options Pane

Control the code created by Simulink® Coder™ code generation software for a Simulink Real-Time™ application. Set up general information about building real-time applications, including target, execution, data logging, and other options.

### Configuration

The **Simulink Real-Time Options** node in the Configuration Parameters dialog box allows you to specify how the software generates the real-time application. To reveal the **Simulink Real-Time Options** node, do the following:

- 1 In the **Code Generation** pane, in the **System target file** list, select one of these settings:
  - `slrt.tlc`  
Generate system target code for Simulink Real-Time.
  - `slrtert.tlc`  
Generate system target code for a Simulink Real-Time using the required Embedded Coder® software.

---

**Note:** If you open a model that was originally saved with **System target file** set to `xpctarget.tlc`, the software updates the setting to `slrt.tlc`, and likewise with `xpctargetert.tlc` and `slrtert.tlc`. To retain the updated setting, save the updated model.

---

- 2 Select **C** for the **Language** parameter on the code generation pane.

### Tips

- The default values work for the generation of most real-time applications. If you want to customize the build of your real-time application, set the option parameters to suit your specifications.
- To access configuration parameters from the MATLAB® command line, use:
  - `gcs` — To access the current model.
  - `set_param` — To set the parameter value.
  - `get_param` — To get the current value of the parameter.

- If you set up your model to Simulink Real-Time Embedded Coder (`slrtert.tlc`), you can create a custom Code Replacement Library (CRL). The CRL must be based on the Simulink Real-Time BLAS (`XPC_BLAS`). For more information, see:
  - “Code Replacement Libraries”
  - “Code You Can Replace From Simulink Models”
  - “Choose a Code Replacement Library”



## Build for default target computer

Direct Simulink Coder to download the real-time application to the default target computer.

### Settings

**Default:** on

On

Downloads the real-time application to the default target computer. Assumes that you configured a default target computer through Simulink Real-Time Explorer.

Off

Enables the **Specify target computer name** field so that you can enter the target computer to which to download the real-time application.

### Dependency

When cleared, this parameter enables **Specify target computer name**.

### Command-Line Information

**Parameter:** xPCisDefaultEnv

**Type:** string

**Value:** 'on' | 'off'

**Default:** 'on'

### See Also

“PCI Bus Ethernet Setup”

“USB-to-Ethernet Setup”

## Specify target computer name

Specify a target computer name for your real-time application.

### Settings

' '

### Tip

The target computer name appears in Simulink Real-Time Explorer as the target computer node, for example `TargetPC1`.

### Dependencies

To enable this parameter, clear **Download to default target computer**.

### Command-Line Information

**Parameter:** `xPCTargetPCEnvName`

**Type:** string

**Value:** Any valid target computer

**Default:** ' '

### See Also

“Simulink Real-Time Explorer Basic Operations”

## Automatically download application after building

Enable Simulink Coder to build and download the real-time application to the target computer.

### Settings

**Default:** on

On

Builds and downloads the real-time application to the target computer.

Off

Builds the real-time application, but does not download it to the target computer.

### Command-Line Information

**Parameter:** xPCisDownloadable

**Type:** string

**Value:** 'on' | 'off'

**Default:** 'on'

### See Also

“Build and Download Real-Time Application”

## Name of Simulink Real-Time object created by build process

Enter the name of the target object created by the build process.

### Settings

**Default:** tg

### Tip

Use this name when you work with the target object through the command-line interface.

### Command-Line Information

**Parameter:** RL320bjectName

**Type:** string

**Value:** 'tg' | valid target object name



**Default:** 'tg'

**See Also**

“Real-Time Application Objects”

## Use default communication timeout

Direct Simulink Real-Time software to wait 5 (default) seconds for the real-time application to be downloaded to the target computer.

### Settings

**Default:** on



Waits the default amount of seconds (5) for the real-time application to be downloaded to the target computer.



Enables the **Specify the communication timeout in seconds** field so that you can enter the maximum length of time in seconds you want to wait for a real-time application to be downloaded to the target computer.

### Dependencies

This parameter enables **Specify the communication timeout in seconds**.

### Command-Line Information

**Parameter:** xPCisModelTimeout

**Type:** string

**Value:** 'on' | 'off'

**Default:** 'on'

### See Also

“Increase the Time for Downloads”

## Specify the communication timeout in seconds

Specify a timeout, in seconds, to wait for the real-time application to download to the target computer.

### Settings

**Default:** 5

### Tip

Enter the maximum length of time in seconds you want the Simulink Real-Time software to wait for the real-time application to download to the target computer. If the real-time application is not downloaded within this time frame, the software generates an error.

### Dependencies

To enable this parameter, set **Use default communication timeout**.

### Command-Line Information

**Parameter:** xPCModelTimeoutSecs

**Type:** string

**Value:** Any valid number of seconds

**Default:** '5'

### See Also

“Increase the Time for Downloads”

## Execution mode

Specify execution mode of downloaded code.

### Settings

**Default:** Real-Time

Real-Time

Executes downloaded code as a real-time application.

Freerun

Executes downloaded code as fast as possible.

Multirate models cannot be executed in Freerun execution mode. On the **Solver** pane in the Configuration Parameters dialog box, set **Tasking mode for periodic sample times** to SingleTasking.

### Command-Line Information

**Parameter:** RL32ModeModifier

**Type:** string

**Value:** 'Real-Time' | 'Freerun'

**Default:** 'Real-Time'

## Real-time interrupt source

Select a real-time interrupt source from the I/O board.

### Settings

**Default:** Timer

Timer

Specifies that the board interrupt source is a timer.

Auto (PCI only)

Enables the Simulink Real-Time software to automatically determine the IRQ that the BIOS assigned to the board and use it.

3 to 15

Specifies that the board interrupt source is an IRQ number on the board.

### Tips

- The Auto (PCI only) option is available only for PCI boards. If you have an ISA board (PC/104 or onboard parallel port), set the IRQ manually.
- The Simulink Real-Time software treats PCI parallel port plug-in boards like ISA boards. For PCI parallel port plug-in boards, set the IRQ manually.
- Multiple boards can share an interrupt number.

### Command-Line Information

**Parameter:** RL32IRQSourceModifier

**Type:** string

**Value:** 'Timer' | Auto (PCI only) | '3' | '4' | '5' | '6' | '7' | '8' | '9' | '10' | '11' | '12' | '13' | '14' | '15'

**Default:** 'Timer'

## **I/O board generating the interrupt**

Specify the board interrupt source.

### **Settings**

**Default:** None/Other

**ATI - RP - R5**

Specifies that the interrupt source is an ATI-RP-R5 board.

**AudioPMC+**

Specifies that the interrupt source is the Bittware AudioPMC+ audio board.

**Bitflow NEON**

Specifies that the interrupt source is the BitFlow™ NEON video board.

**Busmirror EB5100**

Specifies that the interrupt source is the Busmirror EB5100 FlexRay™ board.

**CB\_CIO-CTR05**

Specifies that the interrupt source is the Measurement Computing™ CIO-CTR05 board.

**CB\_PCI-CTR05**

Specifies that the interrupt source is the Measurement Computing PCI-CTR05 board.

**Diamond\_MM-32**

Specifies that the interrupt source is the Diamond Systems MM-32 board.

**FastComm 422/2-PCI**

Specifies that the interrupt source is the Fastcom® 422/2-PCI board.

**FastComm 422/2-PCI-335**

Specifies that the interrupt source is the Fastcom 422/2-PCI-335 board.

**FastComm 422/4-PCI-335**

Specifies that the interrupt source is the Fastcom 422/4-PCI-335 board.

**GE\_Fanuc(VMIC)\_PCI-5565**

Specifies that the interrupt source is the GE® Fanuc VMIC PCI-5565 board.

**General Standards 24DSI12**

Specifies that the interrupt source is the General Standards 24DSI12 board.

**Parallel\_Port**

Specifies that the interrupt source is the parallel port of the target computer.

**Quatech DSCP-200/300**

Specifies that the interrupt source is the Quatech<sup>®</sup> DSCP-200/300 board.

**Quatech ESC-100**

Specifies that the interrupt source is the Quatech ESC-100 board.

**Quatech QSC-100**

Specifies that the interrupt source is the Quatech QSC-100 board.

**Quatech QSC-200/300**

Specifies that the interrupt source is the Quatech QSC-200/300 board.

**RTD\_DM6804**

Specifies that the interrupt source is the Real-Time Devices DM6804 board.

**SBS\_25x0\_ID\_0x100**

Specifies that the interrupt source is an SBS Technologies shared memory board associated with ID 0x100.

**SBS\_25x0\_ID\_0x101**

Specifies that the interrupt source is an SBS Technologies shared memory board associated with ID 0x101.

**SBS\_25x0\_ID\_0x102**

Specifies that the interrupt source is an SBS Technologies shared memory board associated with ID 0x102.

**SBS\_25x0\_ID\_0x103**

Specifies that the interrupt source is an SBS Technologies shared memory board associated with ID 0x103.

**Scramnet\_SC150+**

Specifies that the interrupt source is the Systran<sup>®</sup> Scramnet+ SC150 board.

**Softing\_CAN-AC2-104**

Specifies that the interrupt source is the Softing<sup>®</sup> CAN-AC2-104 board.

**Softing\_CAN-AC2-PCI**

Specifies that the interrupt source is the Softing CAN-AC2-PCI board.

**Speedgoat\_I0301**

Specifies that the interrupt source is the Speedgoat IO301 FPGA board.

**Speedgoat\_I0302**

Specifies that the interrupt source is the Speedgoat IO302 FPGA board.

**Speedgoat\_I0303**

Specifies that the interrupt source is the Speedgoat IO303 FPGA board.

**Speedgoat\_I0311**

Specifies that the interrupt source is the Speedgoat IO311 FPGA board.

**Speedgoat\_I0312**

Specifies that the interrupt source is the Speedgoat IO312 FPGA board.

**Speedgoat\_I0313**

Specifies that the interrupt source is the Speedgoat IO313 FPGA board.

**Speedgoat\_I0314**

Specifies that the interrupt source is the Speedgoat IO314 FPGA board.

**Speedgoat\_I0321**

Specifies that the interrupt source is the Speedgoat IO321 FPGA board.

**Speedgoat\_I0331**

Specifies that the interrupt source is the Speedgoat IO331 FPGA board.

**UEI\_MF<sub>x</sub>**

Specifies that the interrupt source is a United Electronic Industries UEI-MF series board.

**None/Other**

Specifies that the I/O board has no interrupt source.

**Command-Line Information**

**Parameter:** xPCIRQSourceBoard

**Type:** string

**Value:** 'ATI-RP-R5' |  
'AudioPMC+' |  
'Bitflow NEON' |  
'Busmirror EB5100' |  
'CB\_CIO-CTR05' |  
'CB\_PCI-CTR05' |  
'Diamond\_MM-32' |



'FastComm 422/2-PCI' |  
'FastComm 422/2-PCI-335' |  
'FastComm 422/4-PCI-335' |  
'GE\_Fanuc(VMIC)\_PCI-5565' |  
'General Standards 24DSI12' |  
'Parallel\_Port' |  
'Quatech DSCP-200/300' |  
'Quatech ESC-100' |  
'Quatech QSC-100' |  
'Quatech QSC-200/300' |  
'RTD\_DM6804' |  
'SBS\_25x0\_ID\_0x100' |  
'SBS\_25x0\_ID\_0x101' |  
'SBS\_25x0\_ID\_0x102' |  
'SBS\_25x0\_ID\_0x103' |  
'Scramnet\_SC150+' |  
'Softing\_CAN-AC2-104' |  
'Softing\_CAN-AC2-PCI' |  
'Speedgoat\_I0301' |  
'Speedgoat\_I0302' |  
'Speedgoat\_I0303' |  
'Speedgoat\_I0311' |  
'Speedgoat\_I0312' |  
'Speedgoat\_I0313' |  
'Speedgoat\_I0314' |  
'Speedgoat\_I0321' |  
'Speedgoat\_I0331' |  
'UEI\_MFx' |  
'None/Other'  
**Default:** 'None/Other'

## PCI slot (-1: autosearch) or ISA base address

Enter the slot number or base address for the I/O board generating the interrupt.

### Settings

**Default:** -1

The PCI slot can be either -1 (let the Simulink Real-Time software determine the slot number) or of the form [bus, slot].

The base address is a hexadecimal number of the form 0x300.

### Tip

To determine the bus and PCI slot number of the boards in the target computer, in the Command Window, type:

```
tg = slrt;  
getPCIInfo(tg, 'installed')
```

### Command-Line Information

**Parameter:** xPCIOIRQSlot

**Type:** string

**Value:** '-1' | hexadecimal value

**Default:** '-1'

### See Also

“PCI Bus I/O Devices”

## Log Task Execution Time

Log task execution times to the target object property `tg.TETlog`.

For a single-rate model, task execution time (TET) measures how long it takes the kernel to update the model state, propagate the new model state to the outputs, and log the model state during a sample interval. For a multirate model, TET measures how long it takes the kernel to perform those tasks for the base-rate portion only. You can plot the TET to show under what circumstances subsystems are executed and the additional CPU time required for those executions.

### Settings

**Default:** on

On

Logs task execution times to the target object property `tg.TETlog`.

Off

Does not log task execution times to the target object property `tg.TETlog`.

### Command-Line Information

**Parameter:** RL32LogTETModifier

**Type:** string

**Value:** 'on' | 'off'

**Default:** 'on'

### See Also

“Signal Logging Basics”

## Signal logging data buffer size in doubles

Enter the maximum number of sample points to save before wrapping.

### Settings

**Default:** 100000

The maximum value for this option cannot exceed the available target computer memory, which the Simulink Real-Time software also uses to hold other items.

### Tips

- Real-time applications use this buffer to store the time, states, outputs, and task execution time (TET) logs as defined in the Simulink model.
- The maximum value for this option derives from available target computer memory, which the Simulink Real-Time software also uses to hold other items. For example, in addition to signal logging data, the software also uses the target computer memory for the Simulink Real-Time kernel, real-time application, and scopes.

For example, assume that your model has six data items (time, two states, two outputs, and task execution time). If you enter a buffer size of 100000, the target object property `tg.MaxLogSamples` is calculated as `floor(100000 / 6) = 16666`. After the buffer saves 16666 sample points, it wraps and further samples overwrite the older ones.

- Suppose that you enter a logging buffer size larger than the available RAM on the target computer. When you download and initialize the real-time application, the target computer displays a message, **ERROR: allocation of logging memory failed**. To avoid this error, either install more RAM or reduce the buffer size for logging, and then restart the target computer. To calculate the maximum buffer size available for your real-time application logs, divide the amount of available RAM on your target computer by `sizeof(double)`, or 8. Enter that value for the **Signal logging data buffer size in doubles** value.

### Command-Line Information

**Parameter:** RL32LogBufSizeModifier

**Type:** string

**Value:** '100000' | any valid memory size

**Default:** '100000'



## Number of profiling events (each uses 20 bytes)

Enter the maximum of events to log for the profiling tool.

### Settings

**Default:** 5000

The maximum number of events to be logged for the profiling tool.

### Tips

- An event is the start or end of an interrupt or iteration of the model. For example, one sample can have four events: the beginning and end of an interrupt, and the beginning and end of an iteration.
- Each event contains information such as the CPU ID, model thread ID (TID), event ID, and time stamp readings. Each event occupies 20 bytes.

### Command-Line Information

**Parameter:** xPCRL32EventNumber

**Type:** string

**Value:** any valid number of events

**Default:** '5000'

### See Also

“Execution Profiling for Real-Time Applications”

## Double buffer parameter changes

Use a double buffer for parameter tuning. This setting enables parameter tuning so that the process of changing parameters in the real-time application uses a double buffer.

### Settings

**Default:** off

On

Changes parameter tuning to use a double buffer.

Off

Suppresses double buffering of parameter changes in the real-time application.

### Tips

- When a parameter change request is received, the new value is compared to the old one. If the new value is identical to the old one, it is discarded, and if different, it is queued.
- At the start of execution of the next sample of the real-time task, the queued parameters are updated. This operation increases the task execution time (TET) and can cause a CPU overload error.
- Double buffering leads to a more robust parameter tuning interface, but it increases task execution time and the higher probability of overloads. Under typical conditions, keep double buffering off (default).
- If the real-time application contains model parameters, the software ignores this double buffering setting. Normal parameter tuning occurs.

### Command-Line Information

**Parameter:** xpcDb1Buff

**Type:** string

**Value:** 'on' | 'off'

**Default:** 'off'

## Load a parameter set from a file on the designated target file system

Automatically load a parameter set from a file on the designated target computer file system.

### Settings

**Default:** off

On

Enable the automatic loading of a parameter set from the file specified by **File name** on the designated target computer file system.

Off

Suppress the automatic loading of a parameter set from a file on the designated target computer file system.

### Dependencies

This parameter enables **File name**.

### Command-Line Information

**Parameter:** xPCLoadParamSetFile

**Type:** string

**Value:** 'on' | 'off'

**Default:** 'off'

### See Also

“Save and Reload Parameters Using MATLAB Language”



## File name

Specify the target computer file name from which to load the parameter set.

## Settings

' '

## Tip

If the named file does not exist, the software loads the parameter set built with the model.

## Dependencies

To enable this parameter, set **Load a parameter set from a file on the designated target file system**.

## Command-Line Information

**Parameter:** xPCOnTgtParamSetFileName

**Type:** string

**Value:** Any valid file name

**Default:** ' '

## Generate INCA/CANape extensions

Enable real-time applications to generate data, such as A2L data, for Vector CANape® and ETAS® Inca.

### Settings

**Default:** off

On

Enables real-time applications to generate data, such as that for A2L, for Vector CANape and ETAS Inca.

Off

Does not enable real-time applications to generate data, such as that for A2L, for Vector CANape and ETAS Inca.

### Command-Line Information

**Parameter:** xPCGenerateASAP2

**Type:** string

**Value:** 'on' | 'off'

**Default:** 'off'

### See Also

“Prepare ASAP2 Data Description File”

## Enable Stateflow animation

Enables visualization of Stateflow<sup>®</sup> chart animation.

### Settings

**Default:** off

On

Enables visualization of Stateflow chart animation.

Off

Disables visualization of Stateflow chart animation.

### Command-Line Information

**Parameter:** xPCEnableSFAnimation

**Type:** string

**Value:** 'on' | 'off'

**Default:** 'off'

### See Also

“Animate Stateflow Charts Using Simulink External Mode”



# TLC Options Parameters

---

# TLCOptions Properties

Modify real-time application options

## Description

Model options set before code generation to configure the real-time application and the real-time kernel.

To set these options, use the syntax `set_param(model_name, 'TLCOptions', '-aoption_name1=option_value1 -aoption_nameN=option_valueN')`.

Prefix each option name with `-a`. Do not leave spaces around the equals sign. Do not place a comma between consecutive value assignments.

```
set_param(model_name, 'TLCOptions', '-axPCMaxOverloads=20  
-axPCModelStackSizeKB=1024')
```

To read these options, use the syntax `get_param(model_name, 'TLCOptions')`.

```
get_param(model_name, 'TLCOptions')
```

```
ans =
```

```
-axPCMaxOverloads=20 -axPCModelStackSizeKB=1024
```

To remove these options, use the syntax `set_param(model_name, 'TLCOptions', '')`.

```
set_param(model_name, 'TLCOptions', '')
```

## Target Computer Overload

**xPCMaxOverloads** — Number of acceptable target computer overloads

0 (default) | scalar

When `xPCMaxOverloads` is set to a value, such as 3, the Simulink Real-Time software will stop execution with a CPU overload at the following overload (the fourth).

Example: `-axPCMaxOverloads=3`

**xPCMaxOverloadLen** — Number of contiguous acceptable overloads

0 (default) | scalar

You must specify a value that is the same or less than the value for `xPCMaxOverloads`.

When `xPCMaxOverloadLen` is set to a value, such as 2, the software will stop execution with a CPU overload at the following contiguous overload (the third).

Example: `-axPCMaxOverloadLen=2`

**xPCStartupFlag** — Number of executions of the model at startup

1 (default) | scalar

Causes the software to temporarily disable the timer interrupt during model execution. After the model finishes the first `xPCStartupFlag` number of executions, the software reenables the timer interrupt, which invokes the next execution for the model.

Example: `-axPCStartupFlag=3`

## Target Computer Memory

**xPCModelStackSizeKB** — Size of stack memory on the target computer, in kilobytes

512 (default) | scalar

Sets the number of kilobytes of stack memory that are allocated to real-time threads on the target computer

Example: `-axPCModelStackSizeKB=1024`

## Polling Mode

**xpcCPUClockPoll** — Target computer CPU clock rate, in MHz

0 (default) | scalar

Switches the kernel from interrupt mode to polling mode. When **Execution mode is Real-Time**, a nonzero value causes the real-time application to perform a busy wait at the specified polling rate, assumed to be the target computer CPU clock rate. If the value is 0 or if the option is not defined, the kernel executes in interrupt mode.

Example: `-axpcCPUClockPoll=1200`

### Floating-Point Processing

#### **SLRTFTZOFF** — Turns on denormal float processing

0 (default) | scalar

Configures floating-point processing as follows:

- 0 (default) — Denormal float processing is not performed. The representation of extremely small numbers is slightly different from the representation when the value is 1, and floating point operations are faster. The corresponding Microsoft® Visual C++® compiler options are `/fp:fast /arch:SSE2`.
- 1 — Floating point operations meet the IEEE® Standard for Floating Point Arithmetic (IEEE 754-2008). The corresponding Microsoft Visual C++ compiler options are `/fp:precise` with the default value of `/arch`.

Example: `-aSLRTFTZOFF=1`

### More About

- “Maximizing Target Computer CPU Usage”
- “Polling Mode”

### External Websites

- [www.ieee.org](http://www.ieee.org)
- [www.microsoft.com](http://www.microsoft.com)



# Simulink Real-Time Explorer Instrumentation

---

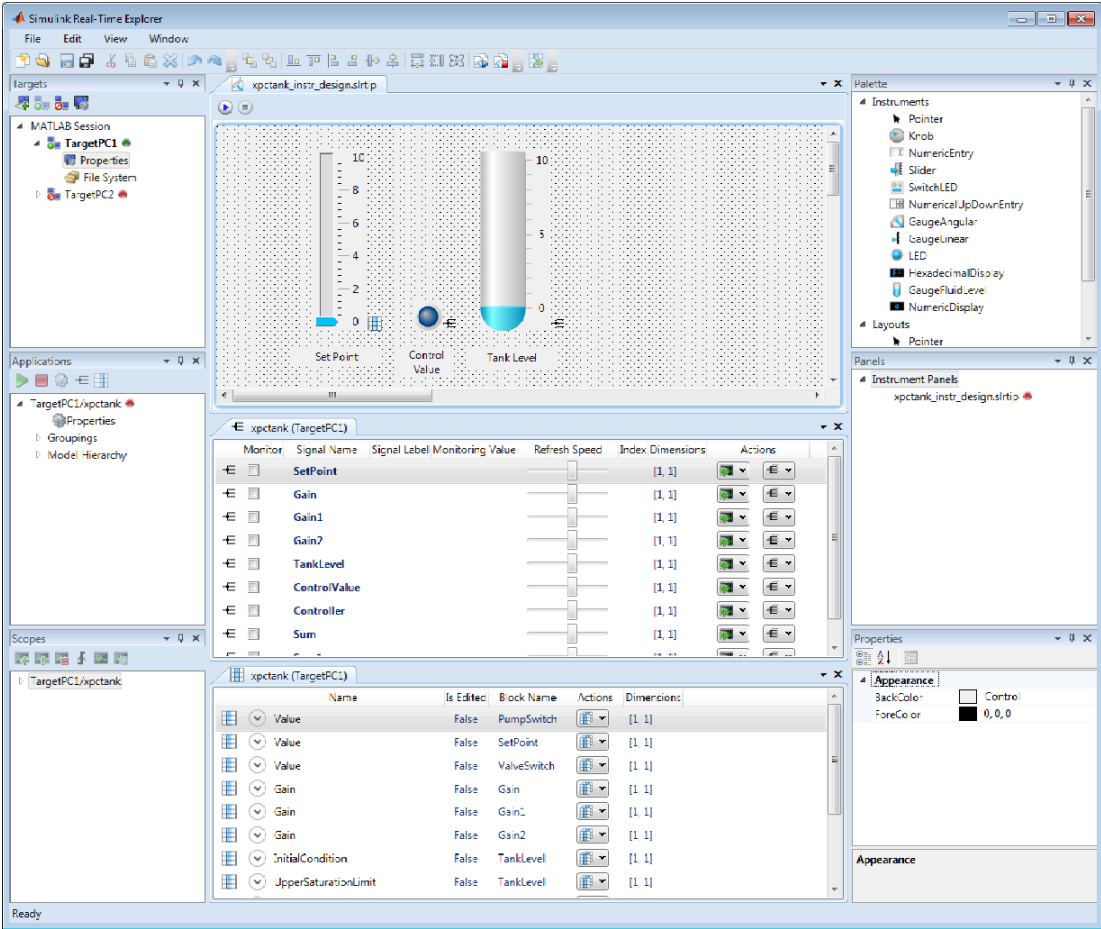
- “Instrumentation for Real-Time Applications” on page 3-2
- “Explorer Configuration Exported to Run Outside MATLAB” on page 3-10
- “Guidelines for Exporting Explorer Configuration” on page 3-12
- “Create Instrument Panel” on page 3-14
- “Configure Instrument for Parameter Tuning” on page 3-15
- “Configure Instruments for Signal Display” on page 3-18
- “Save and Load Environment Properties” on page 3-21
- “Save and Load Instrument Panels” on page 3-22
- “Save and Restore Layouts” on page 3-23
- “Run Instrumented Model” on page 3-24
- “Prepare Explorer Environment for Export” on page 3-27
- “Prepare Instrument Panel Configuration for Export” on page 3-29
- “Export Explorer Configuration” on page 3-31
- “Unpack and Run Standalone Configuration” on page 3-32

## Instrumentation for Real-Time Applications

In this section...
“Instrument Selection and Binding” on page 3-4
“Layout Elements” on page 3-7

To visualize the behavior of a real-time application running on a target computer, Simulink Real-Time Explorer provides instrument panels that you can use. An instrument panel is an Explorer workspace into which you can insert one or more instruments. You can create and load panels from the toolbar, from the File menu, and from the Panels window. You can display simultaneously as many panels as can fit on the screen.

After creating one or more instrument panels, you can drag instruments to the panels and drag parameters and signals to the instruments. You can add layout elements to clarify the relationship between the instruments and the model. You can then start your real-time application from Simulink Real-Time Explorer and start the instrument panels to control the parameters and view the signal outputs.










You can manipulate the instruments using the toolbar buttons.

Action	Icon	Notes
--------	------	-------

New, Open, Save, SaveAll		
--------------------------	--	--

The available operations vary with the active window.

- You can save an instrument panel from within only the window for that panel.


Action	Icon	Notes
		<ul style="list-style-type: none"> <li>You can create or open a signal or parameter group from within only the <b>Applications</b> window.</li> <li>You can save a group from within only the window for that group.</li> <li>You can create or open an instrument panel regardless of active window.</li> <li>You can use the <b>SaveAll</b> button regardless of active window.</li> </ul>
Cut, Copy, Paste, Delete		Applies to instruments only.
Layer		Applies to instruments only.
Align on edges		Applies to instruments only.
Align on centers		Applies to instruments only.
Equalize sizes		Applies to instruments only.
Undo, Redo		Available after unsaved change only.
Run, Stop, Run all, Stop all		Run and stop all active instrument panels.


## Instrument Selection and Binding

To instrument a model, populate an instrument panel with instruments compatible with the parameters and signals that they represent in the model.

To make an instrument interact with the real-time application, bind a signal or parameter to the instrument. You bind the signal or parameter by dragging it from the signal or parameter viewer to the instrument. You can also drag a signal or parameter from a signal or parameter group to an instrument.

You can bind a signal to a signal display instrument but not to a parameter tuning instrument. You can bind a parameter to a parameter tuning instrument and to a signal display instrument.

When you bind a signal to a signal display instrument, the **Signal** button  appears next to the instrument.

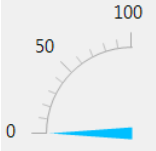
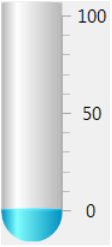
When you bind a parameter to a parameter tuning or signal display instrument, the **Parameter** button  appears next to the instrument.

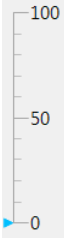



When the instrument panel is running, you can tune parameters using parameter tuning instruments. The software transmits the parameter changes to the real-time application. You can view the changed behavior using signal display instruments.

When the instrument panel is not running, you can add, remove, and lay out instruments and connect signals and parameters to them.

**Signal and Parameter Display Instruments**

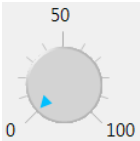
The table shows some of the tasks you can do with signal and parameter display instruments.

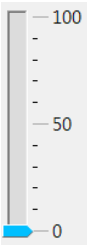
To	Requiring	Use
<ul style="list-style-type: none"> <li>• Show the pressure in a container.</li> <li>• Show the speed of a vehicle.</li> <li>• Show current or voltage in a circuit.</li> </ul>	<ul style="list-style-type: none"> <li>• Real-valued data</li> <li>• Analogy to physical instruments that show approximate value by angular displacement</li> </ul>	 <p>GaugeAngular</p>
<ul style="list-style-type: none"> <li>• Show the level of fluid in a container.</li> <li>• Show the pressure in a pipe.</li> </ul>	<ul style="list-style-type: none"> <li>• Real-valued data</li> <li>• Analogy to physical instruments that show approximate value by vertical displacement</li> </ul>	 <p>GaugeFluidLevel</p>

To	Requiring	Use
<ul style="list-style-type: none"> <li>Show the pressure in a container.</li> <li>Show audio output power.</li> <li>Show current or voltage in a circuit.</li> </ul>	<ul style="list-style-type: none"> <li>Real-valued data</li> <li>Analogy to physical instruments that show approximate value by linear displacement</li> </ul>	 <p>GaugeLinear</p>
<ul style="list-style-type: none"> <li>Show traffic on a digital bus.</li> <li>Show the state of a state machine.</li> <li>Show on-off state of a switch.</li> <li>Show on-off state of a bidirectional pin.</li> </ul>	<ul style="list-style-type: none"> <li>Integer-valued data</li> <li>Analogy to physical instruments that show hexadecimal values</li> <li>Boolean data</li> <li>Analogy to physical instruments that show value by lights turning on and off</li> </ul>	 <p>HexadecimalDisplay</p>  <p>LED</p>
<ul style="list-style-type: none"> <li>Show a temperature measurement to given precision.</li> <li>Show a voltage measurement to given precision.</li> <li>Show a date and time.</li> </ul>	<ul style="list-style-type: none"> <li>Real-valued data</li> <li>Analogy to physical instruments that show real values in decimal and other format.</li> </ul>	 <p>NumericDisplay</p>

#### Parameter Tuning Instruments

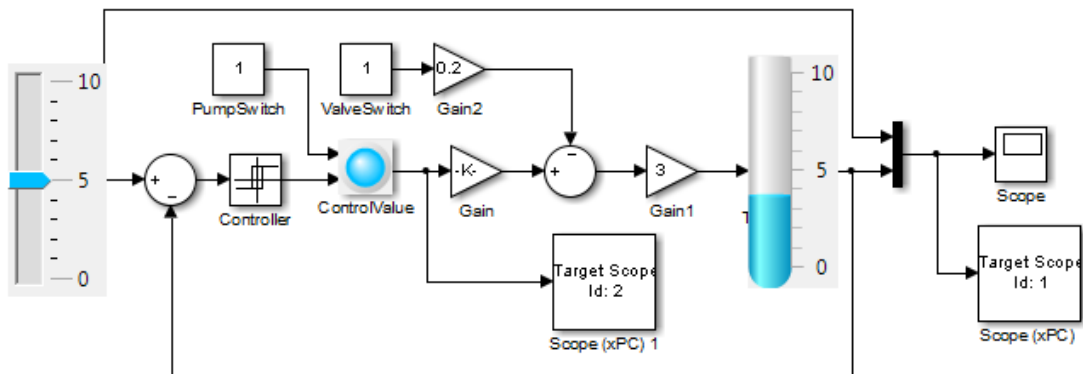
The table shows some of the tasks you can do with parameter tuning instruments.

To	Requiring	Use
<ul style="list-style-type: none"> <li>Control gain of a radio receiver.</li> <li>Control amplification of a radio transmitter.</li> </ul>	<ul style="list-style-type: none"> <li>Real-valued data</li> <li>Analogy to physical controls that set</li> </ul>	

To	Requiring	Use
	approximate value by angular displacement	Knob
<ul style="list-style-type: none"> <li>Enter initial set point control value for thermostat.</li> <li>Enter seed value for random number generator.</li> </ul>	<ul style="list-style-type: none"> <li>Real-valued data</li> <li>Analogy to physical controls that set exact numeric value</li> </ul>	<input type="text" value="0"/> NumericEntry
<ul style="list-style-type: none"> <li>Control car radio audio volume using step-increment.</li> <li>Smoke test controller range in small number of clicks.</li> </ul>	<ul style="list-style-type: none"> <li>Real-valued data</li> <li>Analogy to physical controls that set initial value and step increment</li> </ul>	<input type="text" value="0"/> <input type="button" value="↑"/> <input type="button" value="↓"/> NumericUpDownEntry
<ul style="list-style-type: none"> <li>Control frequency of a radio receiver.</li> <li>Control pressure valve setting.</li> </ul>	<ul style="list-style-type: none"> <li>Real-valued data</li> <li>Analogy to physical controls that set approximate values by linear displacement</li> </ul>	 Slider
<ul style="list-style-type: none"> <li>Turn on a power supply.</li> <li>Close a gate valve.</li> </ul>	<ul style="list-style-type: none"> <li>Boolean data</li> <li>Analogy to physical controls that are either on or off</li> </ul>	<input type="checkbox" value="ON"/> SwitchLED

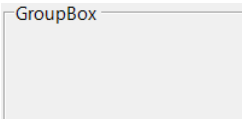
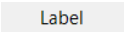

### Layout Elements

To make the relationship between the instruments and the model clearer, you can add layout elements, such as a picture box. The figure shows a picture box that shows which signals the instruments are bound to.




Tank Level Control System

The table shows some ways that you can modify your layout with layout elements.

To	Requiring	Use
<ul style="list-style-type: none"> <li>Group electrical instruments together and label the group.</li> </ul>	<ul style="list-style-type: none"> <li>Design-time box resizing</li> <li>Run-time static display</li> </ul>	 <p>GroupBox</p>
<ul style="list-style-type: none"> <li>Group pressure instruments together and label the group.</li> </ul>		GroupBox
<ul style="list-style-type: none"> <li>Label an electrical instrument.</li> </ul>	<ul style="list-style-type: none"> <li>Design-time box resizing, left-right and up-down text alignment</li> </ul>	 <p>Label</p>
<ul style="list-style-type: none"> <li>Label a pressure instrument.</li> </ul>	<ul style="list-style-type: none"> <li>Run-time static display</li> </ul>	Label
<ul style="list-style-type: none"> <li>Group the electrical group with the pressure group and scroll between the groups.</li> </ul>	<ul style="list-style-type: none"> <li>Design-time box resizing</li> <li>Run-time box scrolling</li> </ul>	
<ul style="list-style-type: none"> <li>Group a picture box with a group of instruments to show what the instruments are measuring.</li> </ul>		Panel



To	Requiring	Use
<ul style="list-style-type: none"><li>• Insert an image of a circuit diagram on a panel behind electrical instruments.</li><li>• Insert an image of a circulation diagram on a panel behind pressure instruments.</li></ul>	<ul style="list-style-type: none"><li>• Design-time image stretch, zoom, center, and autosize</li><li>• Run-time static display</li></ul>	 PictureBox

## Explorer Configuration Exported to Run Outside MATLAB

You can export a Simulink Real-Time Explorer configuration as a standalone executable to run outside MATLAB on a computer compatible with Windows®. The standalone Explorer supports a subset of the capabilities that it supports under MATLAB.

- You cannot change the communication parameters that the interface uses to communicate with the target computer. Before you export the Simulink Real-Time Explorer configuration, configure and test the communication parameters.

To access more than one target computer, in the **Targets** window, configure a separate **Session** record for each target computer.

- For each instrument, the exporting software records the real-time application and target computer environment with which it is associated. To interact with multiple target computers, create separate instrument panels for each separate real-time application and target computer combination.
- If you rename a target computer, update the **TargetName** parameter for each associated instrument to maintain the connection to the real-time application.
- You cannot load or unload a real-time application from the standalone executable. Before you start the executable, start the real-time application on the target computer.
- You can access only instrument panels and windows that you loaded before you exported the configuration.
- You cannot access the real-time application model hierarchy from the standalone executable.
- You can access only signals in signal groups that you loaded before you exported the configuration.
- You cannot move a signal from one signal group to another group, or create or load a new signal group.
- You can access only parameters in parameter groups that you loaded before you exported the configuration.
- You cannot move a parameter from one parameter group to another group, or create or load a new parameter group.
- You cannot save session layouts. If you close a window, you can restore the original layout using **File > Restore Original View**.

## **Related Examples**

- “Standalone Boot Method”

## **More About**

- “Guidelines for Exporting Explorer Configuration” on page 3-12

# Guidelines for Exporting Explorer Configuration

In this section...
“Execution Environment” on page 3-12
“Signal Groups” on page 3-13
“Parameter Groups” on page 3-13
“Instrument Panels” on page 3-13
“Window Layout” on page 3-13

Before exporting a Simulink Real-Time Explorer configuration, set up the execution environment, signal and parameter groups, and panels. Lay out the windows the way that you want them in the standalone executable. Follow these guidelines:

## Execution Environment

For each computer on which you intend to run the standalone Simulink Real-Time Explorer executable:

- Verify that the candidate computer is compatible with 64-bit Windows.
- Verify that the CPU and operating system meet the requirements for executing the standalone Simulink Real-Time Explorer executable.
- Verify that Microsoft .NET Framework 4.5 is installed on the candidate computer.

For each target computer on which you intend to run the real-time application:

- Verify that the target computer meets the requirements for running the real-time application.
- In Simulink Real-Time Explorer, configure the target and communication settings to connect each computer that is compatible with Windows to each target computer.

You can have only one target computer node for each unique **IP address** setting.

- Configure the **Boot mode** setting for each target computer as **Stand Alone**.
- Optionally, rename the target computer session from **TargetPCx** to something more specific to your system.
- As a test, build and download a real-time application to each target computer connected to the development computer running Simulink Real-Time Explorer.

The real-time application on the target computer is the same application that you intend to access with the standalone executable.

## Signal Groups

- To access signals, add them from the model hierarchy to a signal group.
- To include a signal group in the standalone package, load it into the current session.

## Parameter Groups

- To access parameters, add them from the model hierarchy to a parameter group.
- To include a parameter in the standalone package, load it into the current session.

## Instrument Panels

- To interact with multiple target computers, create a separate instrument panel for each separate real-time application and target computer pair.
- To include an instrument panel in the standalone package, load it into the current Simulink Real-Time Explorer session.
- If you renamed the target computer, update the **TargetName** parameter for each instrument to maintain the connection to the real-time application.

## Window Layout

- You can configure which windows the software opens on startup by opening the windows and arranging them accordingly. When you export the model configuration, the software includes the windows layout in the standalone package.
- To return to the initial standalone executable layout, click **File > Restore Initial View**.

## Related Examples

- “Standalone Boot Method”


## More About

- “Explorer Configuration Exported to Run Outside MATLAB” on page 3-10

### Create Instrument Panel

In this step, you create and save an instrument panel for the `xpctank` model. Start by building and downloading the real-time application to the target computer, running Simulink Real-Time Explorer, and connecting Explorer to the target computer.

To create an instrument panel:

- 1 In the **Panels** pane, right-click the **Instrument Panels** node, and then click **Add New**.
- 2 Type a name and folder in the **Name** and **Location** text boxes. Give the panel a name like `xpctank_instr_design.slrtip`, and then press **Enter**.
- 3 Click the **Save** button .

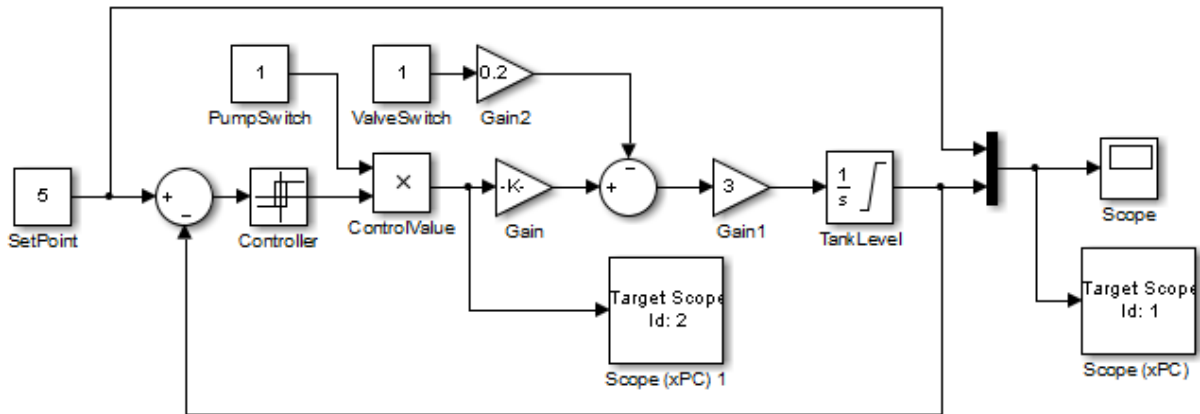
The next task is “Configure Instrument for Parameter Tuning” on page 3-15.

#### Related Examples

- “Save and Restore Layouts” on page 3-23

## Configure Instrument for Parameter Tuning

In this step, you select and configure an instrument to tune a parameter in the `xpctank` model. You must have previously created the `xpctank_instr_design.slrtip` instrument panel (see “Create Instrument Panel” on page 3-14).



Tank Level Control System

The parameter characteristics are listed in this table.

Name	Type	Range	Purpose
SetPoint	Numeric	0–10 units	Represents the level at which the controller maintains the tank fluid level. You do not have to set it to an exact value.

The **Slider** instrument meets the requirement for **SetPoint**. To set an exact numeric value, use, for example, a **NumericEntry** instrument.

To select and configure the instrument:



- 1 Load the instrument panel.

In the **Panels** pane, right-click the **Instrument Panels** node and select **Existing**. From the list, select `xpctank_instr_design.slrtip`.

- 2 Select the instrument.

From the **Palette** pane, drag a **Slider** instrument to the `xpctank_instr_design.slrtip` instrument panel.

- 3 Bind the parameter to the instrument.

To bind the **SetPoint** parameter to the **Slider** instrument, open the **Parameter** workspace for model `xpctank` ( on the toolbar). Drag the **Parameter** icon  next to parameter **SetPoint** to the **Slider** instrument.

A small copy of the **Parameter** icon appears next to the **Slider** instrument.

- 4 Set the instrument range.

Click the **Slider** instrument, and then click the **Tasks** button  in the top right corner.

- 5 In the **Slider Tasks** dialog box, set property **Min** to 0 and property **Span** to 10.

- 6 Select and configure a label.

From the **Palette** pane, drag a **Label** layout item to under the **Slider** instrument.

- 7 Click the **Label** element.

- 8 In the **Properties** pane, scroll down to the **Appearance** node. Set the **Text** property to **Set Point**, and then press **Enter**.

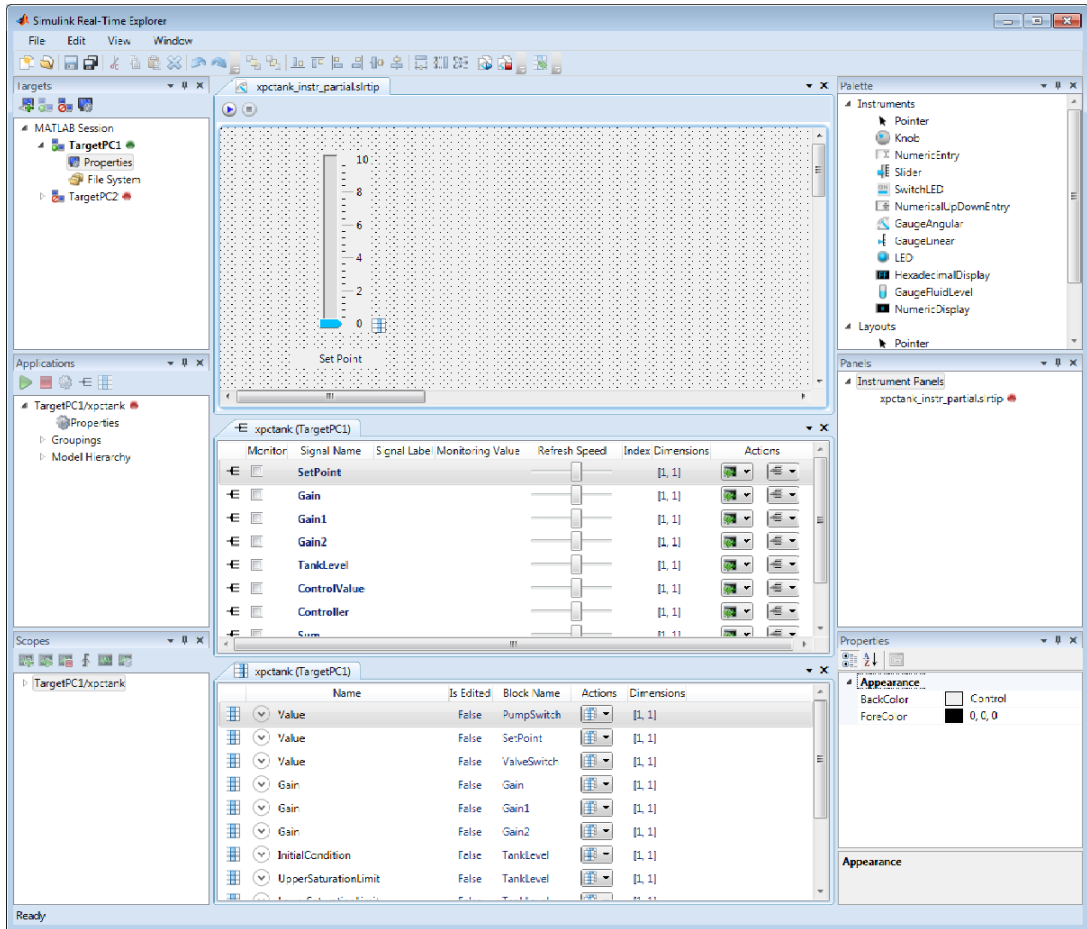
- 9 Scroll down to the **TextAlign** property. Click the down arrow and click the center of the nine blocks presented.

The **TextAlign** property becomes **MiddleCenter**.

- 10 Click the **Save** button .

At the end of this task, Simulink Real-Time Explorer looks like this figure.





The next task is “Configure Instruments for Signal Display” on page 3-18.

## Related Examples

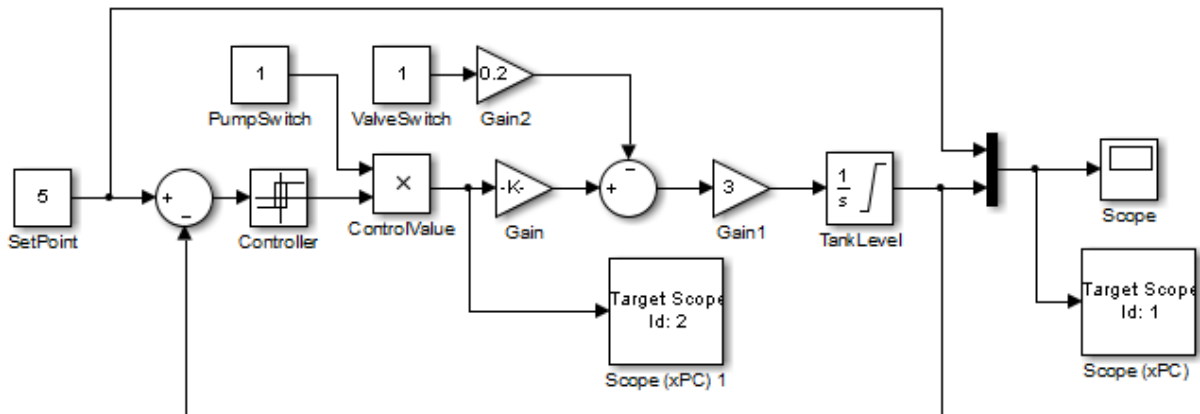
- “Save and Restore Layouts” on page 3-23

## More About

- “Instrumentation for Real-Time Applications” on page 3-2

## Configure Instruments for Signal Display

In this step, you select and configure instruments to display two signals in the xpctank model. You must have previously created the xpctank\_instr\_design.slrtip instrument panel (see “Create Instrument Panel” on page 3-14).



Tank Level Control System

The signal characteristics are listed in this table.

Name	Type	Range	Purpose
TankLevel	Numeric	0–10 units	Represents the current tank fluid level. You do not have to display an exact value.
ControlValue	Boolean	1, 0	Represents the state of the pump (on or off).

- The GaugeFluidLevel instrument meets the requirements for TankLevel. To display an exact numeric value, use, for example, a NumericDisplay instrument.
- The LED instrument meets the requirements for ControlValue.

To select and configure each instrument:

- 1 Load the instrument panel.

In the **Panels** pane, right-click the **Instrument Panels** node and select **Existing**. From the list, select `xpctank_instr_design.slrtip`.

- 2 Select the instrument.

From the **Palette** pane, drag a `GaugeFluidLevel` instrument to the `xpctank_instr_design.slrtip` instrument panel.

- 3 Bind the signal to the instrument.

To bind the `TankLevel` signal to the `GaugeFluidLevel` instrument, open the Signal workspace for model `xpctank` (☰ on the toolbar). Drag the **Signal** icon ☰ next to signal `TankLevel` to the `GaugeFluidLevel` instrument.

A small copy of the **Signal** icon appears next to the `GaugeFluidLevel` instrument.

- 4 Set the instrument range as required.

Select the `GaugeFluidLevel` instrument, and then click the **Tasks** button (▶) in the top right corner.

- 5 In the **GaugeFluidLevel Tasks** dialog box, set property **Min** to 0 and property **Span** to 10.

- 6 Select and configure a label.

From the **Palette** pane, drag a `Label` layout item to under the `GaugeFluidLevel` instrument.

- 7 Click the `Label` element.

- 8 In the **Properties** pane, scroll down to the **Appearance** node. Set the **Text** property to `Tank Level`, and then press **Enter**.

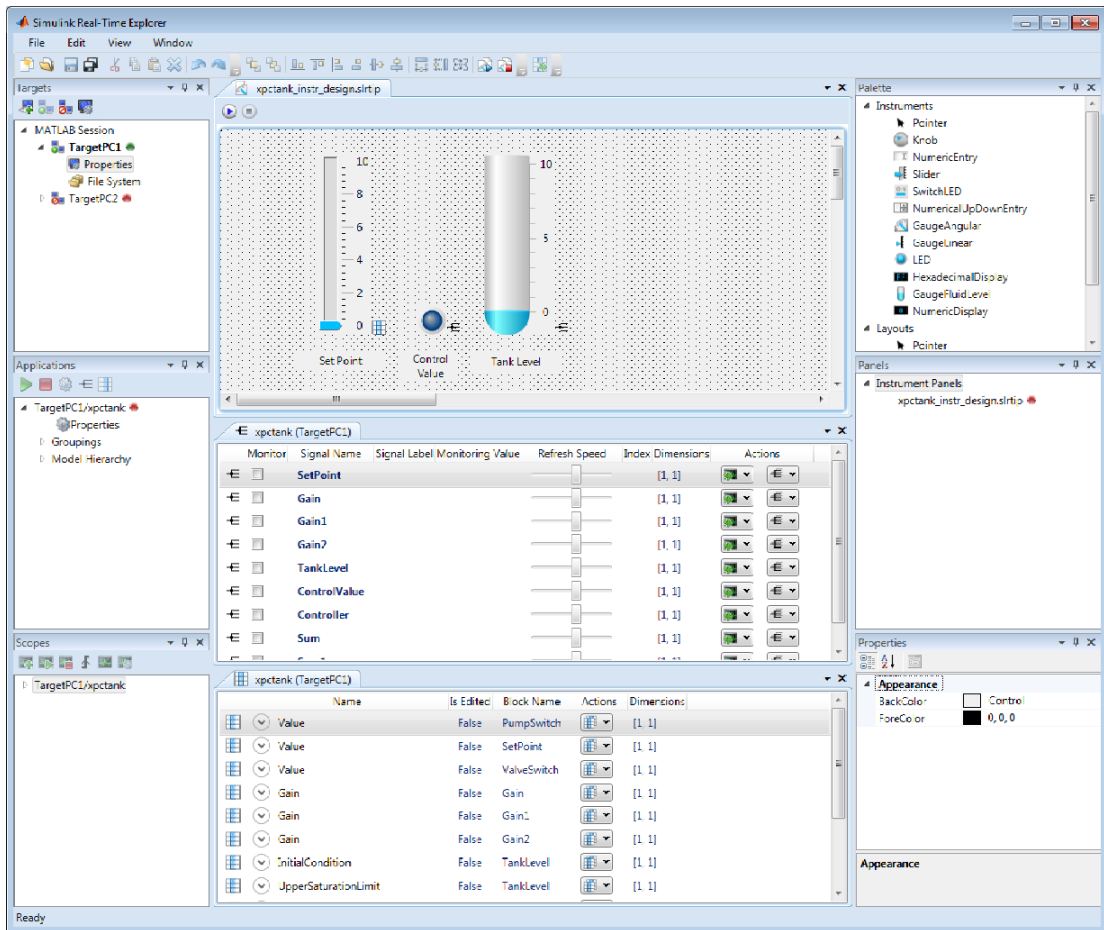
- 9 Scroll down to the **TextAlign** property. Click the down arrow and click the center of the nine blocks presented.

The **TextAlign** property becomes `MiddleCenter`.

- 10 Click the **Save** button .

Using a similar procedure, add an LED instrument to the instrument panel and bind signal `ControlValue` to it. Label the LED `Control Value`.

At the end of this task, Simulink Real-Time Explorer looks like this figure.



The next task is “Run Instrumented Model” on page 3-24.

## Related Examples

- “Save and Restore Layouts” on page 3-23

## More About


- “Instrumentation for Real-Time Applications” on page 3-2

## Save and Load Environment Properties

The Simulink Real-Time Explorer environment consists of the property settings you define for the **Targets** pane. You can save your settings for the next session.

- 1 Set properties in the **Targets** pane.

After you change one or more properties and press **Enter**, the **Save** button and menu item are available.


- 2 To save your environment properties, click the **Save** button  in the toolbar.

If you do not explicitly save the environment settings, Simulink Real-Time Explorer asks on exit if you want to save them.

When starting, Simulink Real-Time Explorer loads the last-saved set of environment properties.

## Save and Load Instrument Panels

As you are developing instrument panels to control your model, you can save your panels or load existing panels. You can load more than one instrument panel at a time.

- To save your instrument panel, click in the **Panels** pane, and then click the **Save** button .
- To load an existing instrument panel, in the **Panels** pane, right-click the **Instrument Panels** node and select **Existing**.

## Save and Restore Layouts

As you are configuring Simulink Real-Time session layouts, you can save your layout or restore a previous layout. Saving a layout preserves the following information:

- The position of the open panes and tabs.
- The target computer connections.
- The instrument panels that you loaded.
- The signal and parameter groups that you loaded.

Saving a layout saves the **Scopes** pane position, but it does not save the state of the scopes in the pane. In particular, if you add a scope within a Simulink Real-Time session, the software does not restore the new scope with the rest of the layout.




- To save a session layout, click **File > Save Layout**.
- To restore a session layout, click **File > Restore Layout**.

## Run Instrumented Model

This example shows how to run the instrumented `xpctank` model. Before carrying out this procedure, you must have performed the steps in “Configure Instrument for Parameter Tuning” on page 3-15 and “Configure Instruments for Signal Display” on page 3-18.

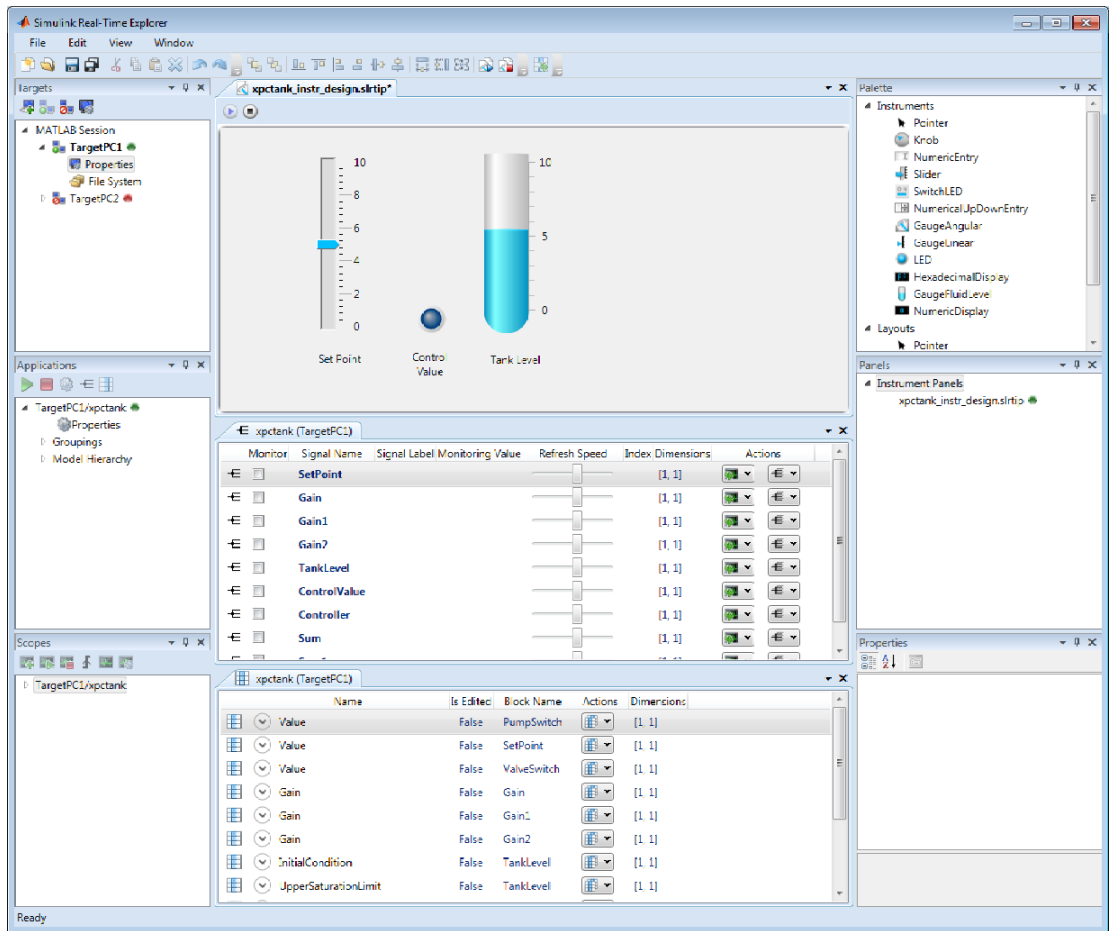
- 1 Load the instrument panel.



In the **Panels** pane, right-click the **Instrument Panels** node and select **Existing**. From the list, select `xpctank_instr_design.slrtip`.

- 2 Set property **Stop time** to `inf` in the **Applications** pane ( on the toolbar).
- 3 To start the instrument, in the `xpctank_instr_design.slrtip` instrument panel, click the **Run Instrument** button .
- 4 To start execution, in the **Applications** pane, click the real-time application, and then click the **Start** button  on the toolbar.
- 5 Using the **Slider** instrument, set the tank level to the required value, such as 5.

The tank level rises to and oscillates around the set point, as shown in this figure.





- 6 To stop execution, in the **Applications** pane, click the real-time application, and then click the **Stop** button  on the toolbar.
- 7 To stop the instruments, in the xpctank\_instr\_design.slrtip instrument panel, click the **Stop Instrument** button .

## Related Examples

- “Real-Time Application Instruments”

### **More About**

- “Instrumentation for Real-Time Applications” on page 3-2

## Prepare Explorer Environment for Export

Verify that each combination of a candidate computer that is compatible with Windows and a target computer works together. Each target computer must run in standalone mode.

The example uses the instrumented `xpctank` model. Before carrying out this procedure, you must have performed the steps in “Real-Time Application Instruments”.

For each computer on which you intend to run the standalone Simulink Real-Time Explorer executable:

- 1 Verify that the candidate computer is compatible with 64-bit Windows.
- 2 Verify that the CPU and operating system meet the requirements for executing the standalone Simulink Real-Time Explorer executable.
- 3 Verify that Microsoft .NET Framework 4.5 is installed on the candidate computer.

For each target computer on which you intend to run the real-time application:

- 1 Verify that the target computer CPU and operating system meet the requirements for running the Simulink Real-Time kernel.
- 2 Verify that the Simulink Real-Time Explorer **Targets** pane contains a target computer node representing each target computer that you intend to access.

If you rename a target computer node, make the corresponding change in the **Bindings > TargetName** property for each instrument.

- 3 Verify that the settings in the **Host-to-Target communication** tab match the requirements of the target computer.

You can have only one target computer node for each unique **IP address** setting.

- 4 Verify that the settings in the **Target settings** tab match the capabilities of the target computer.
- 5 Prepare and copy the required kernel and real-time application files to the target computer. In the **Boot configuration** tab, set **Boot mode** to **Stand Alone**.
- 6 Connect the target computer to the candidate computer and restart the target computer. Verify that the target computer loads the Simulink Real-Time kernel and starts the real-time application.

### **Related Examples**

- “Development Computer Requirements”
- “Target Computer Requirements”
- “PCI Bus Ethernet Setup”
- “USB-to-Ethernet Setup”
- “Target Computer Settings”
- “Standalone Boot Method”

### **More About**

- “Explorer Configuration Exported to Run Outside MATLAB” on page 3-10
- “Guidelines for Exporting Explorer Configuration” on page 3-12

## Prepare Instrument Panel Configuration for Export

Load the instrument panels for the instrumented model. Resize and lay out Simulink Real-Time Explorer.

The example uses the instrumented `xpctank` model.

---

**Note:** When you run the standalone executable, you cannot access the model hierarchy. You can access only instrument panels and windows that were open when you exported the configuration. You can access only signals and parameters that were loaded in signal and parameter groups when you exported the configuration.

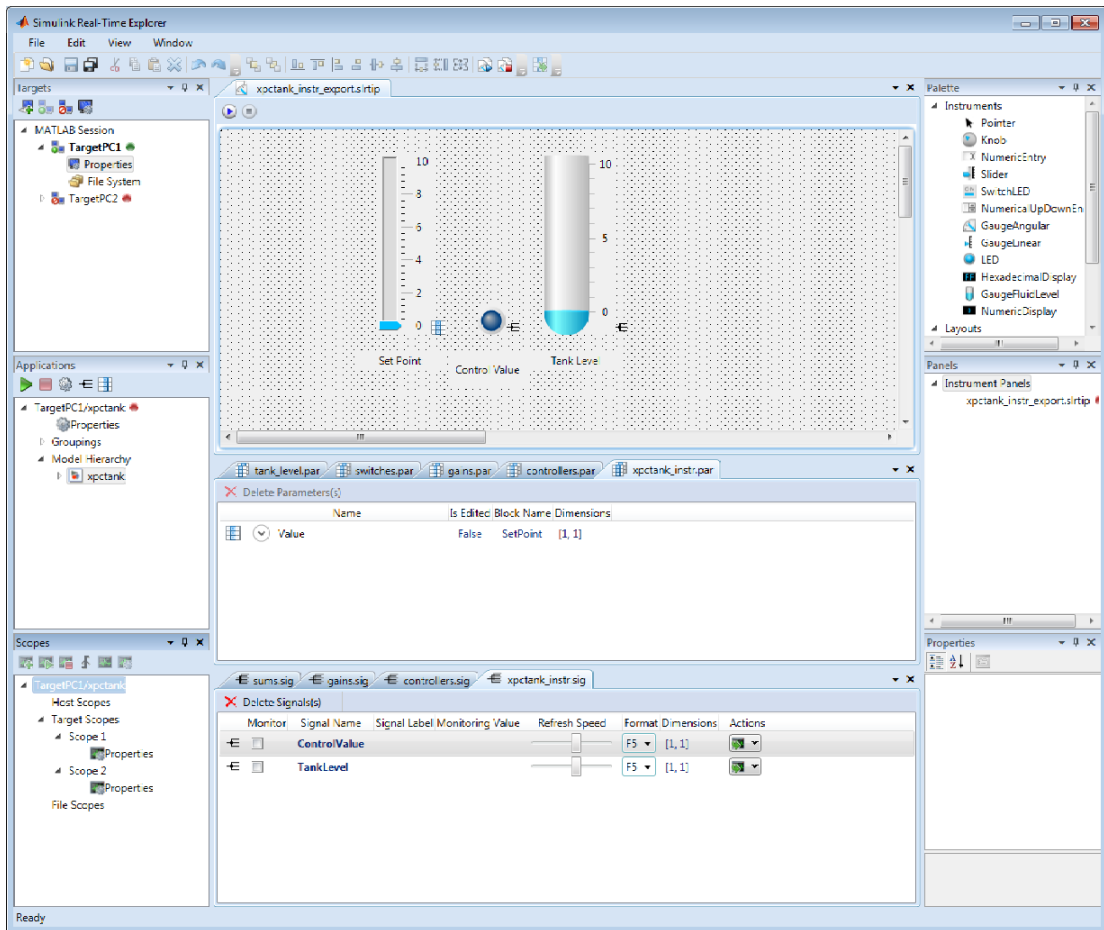
---

- 1 Load your instrument panels into Simulink Real-Time Explorer.

Here, the panel is `xpctank_instr_design.slrtip`.

- 2 Create a parameter group for the key block representing the set point (`xpctank_instr.par`).
- 3 Create parameter groups for the low-level blocks representing the tank level, switches, gains, and controllers (`tank_level.par`, `switches.par`, `gains.par`, and `controllers.par`).
- 4 Create a signal group for the key blocks representing the control value and tank level (`xpctank_instr.sig`).
- 5 Create signal groups for the low-level blocks representing the sums, gains, and controllers (`sums.sig`, `gains.sig`, and `controllers.sig`).
- 6 Open, lay out, and resize the windows that you want the standalone executable to open.
- 7 Save each instrument panel.

The `xpctank_instr_design.slrtip` configuration looks like the figure.



The next task is “Export Explorer Configuration” on page 3-31.

## Related Examples

- “Create Parameter Groups Using Simulink Real-Time Explorer”
- “Create Signal Groups Using Simulink Real-Time Explorer”

## More About

- “Instrumentation for Real-Time Applications” on page 3-2

## Export Explorer Configuration

Export the Simulink Real-Time Explorer configuration as a standalone executable.

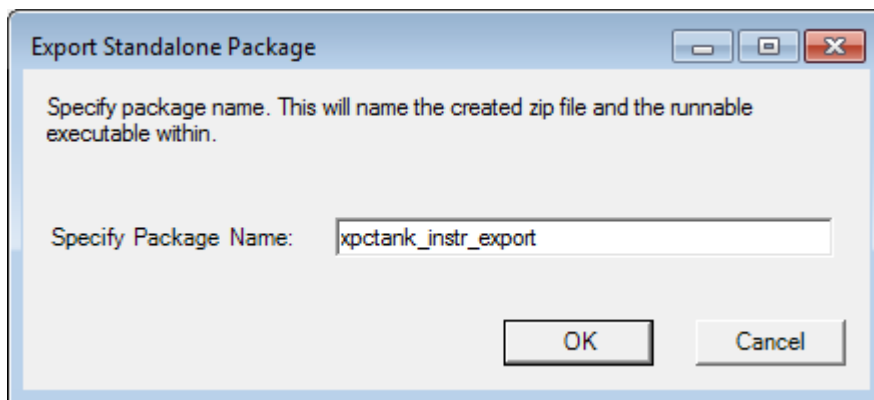
The example uses the instrumented `xpctank` model. Before carrying out this procedure, you must have performed the steps in “Prepare Instrument Panel Configuration for Export” on page 3-29.

---

**Note:** When you run the standalone executable, you cannot access the model hierarchy. You can access only instrument panels and windows that were open when you exported the configuration. You can access only signals and parameters that were loaded in signal and parameter groups when you exported the configuration.

---

- 1 To export the configuration as a standalone executable, click **File > Export**.
- 2 In the **Specify Package Name** text box, type `xpctank_instr_export`.



- 3 Click **OK**.

The software generates a file named `xpctank_instr_export.zip` in the current folder.

The next task is “Unpack and Run Standalone Configuration” on page 3-32.

### More About

- “Instrumentation for Real-Time Applications” on page 3-2

## Unpack and Run Standalone Configuration

Unpack the standalone executable onto a computer that is compatible with Windows.

The example uses the instrumented `xpctank` model. Before carrying out this procedure, you must have performed the steps in “Export Explorer Configuration” on page 3-31.

---

**Note:** When you run the standalone executable, you cannot access the model hierarchy. You can access only instrument panels and windows that were open when you exported the configuration. You can access only signals and parameters that were loaded in signal and parameter groups when you exported the configuration.

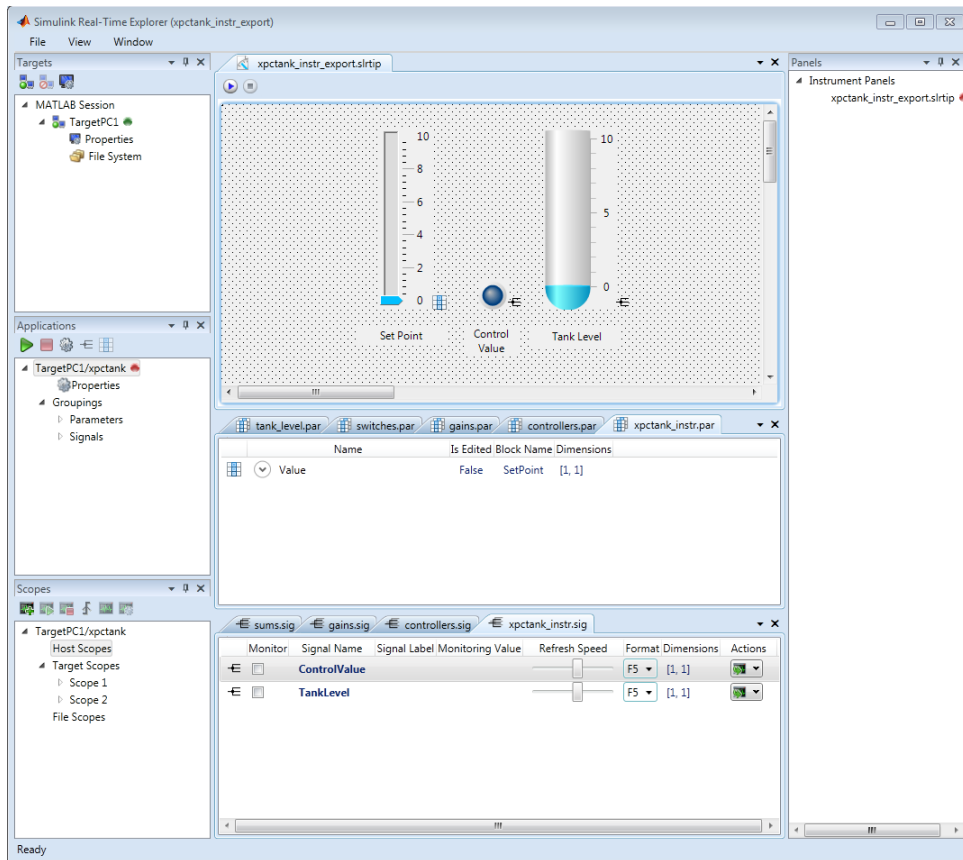
---

- 1 Copy `xpctank_instr_export.zip` from the original folder to a folder on the computer compatible with Windows, for example `C:\workdir`.
- 2 Double-click `xpctank_instr_export.zip`.
- 3 In the unzip program dialog box, click **Extract**.
- 4 Select the extraction root folder, and then click **Extract**.
- 5 Navigate to folder `xpctank_instr_export` in the extraction root folder.
- 6 Connect the target computer to the computer that is compatible with Windows. Restart the target computer.

Verify that the target computer loads the Simulink Real-Time kernel and the real-time application.

- 7 Double-click `runxpctank_instr_export.exe`.





To interact with the real-time application on the target computer, use the executable interface.

- If a signal is accessible, you can add a scope and attach the signal to the scope. Scopes that you add and remove do not change the model.
- If you remove a window, you can restore it by clicking **File > Restore Original View**.

## More About

- “Instrumentation for Real-Time Applications” on page 3-2



# Simulink Real-Time Explorer Instruments

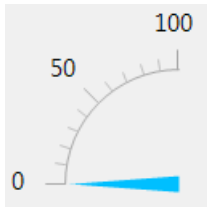
---

GaugeAngular  
GaugeFluidLevel  
GaugeLinear  
GroupBox  
HexadecimalDisplay  
Knob  
Label  
LED  
NumericDisplay  
NumericEntry  
NumericUpDownEntry  
Panel  
PictureBox  
Slider  
SwitchLED

# GaugeAngular

Graphic instrument to display signal values



## Description



Use the `GaugeAngular` instrument to display real-valued data suitable for an angular gauge, such as pressure, speed, and current.

## Key Parameters

The key parameters are under the **Instrument** node in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

## Scale Graphic Display

The root node of this parameter is **Instrument**.

Parameter	Usage
<code>AutoSize</code>	If <code>True</code> , size the graphic to accommodate the parts of the display

The root node of these parameters is **Instrument+ScaleDisplay+GeneratorAuto**.

Parameter	Usage
-----------	-------

<b>DesiredIncrement</b>	Display of major tick values. number of labels = span/(desired increment + 1). Does nothing if the required labels do not fit in the space available in the graphic.
<b>FixedMinMaxMajor</b>	If True, the top and bottom ticks are constrained to be major ticks with min/max values defined by <b>Min</b> and <b>Span</b>
<b>MidIncluded</b>	If True, insert a tick halfway between major ticks.  If <b>MinorCount</b> is even, space the minor ticks equally around the center tick. If <b>MinorCount</b> is odd, replace the center tick with the middle tick. If
<b>MinorCount</b>	Number of minor ticks between major ticks
<b>MinTextSpacing</b>	Minimum space between scale ticks

## Scale Text Display

The root node of these parameters is **Instrument+ScaleDisplay+TextFormatting**.

Parameter	Usage
<b>Precision</b>	Number of digits to the right of the decimal point
<b>PrecisionStyle</b>	One of the values FixedDecimalPoints, SignificantDigits, None
<b>Style</b>	One of the values Number, Thousands, Prefix, Exponent, Price32nds, DateTime, DateTimeUTC
<b>UnitsText</b>	Display unit next to tick labels

## General Scale Range

The root node of these parameters is **Instrument+ScaleRange**.

<b>Parameter</b>	<b>Usage</b>
<b>Min</b>	Minimum possible value
<b>Reverse</b>	If True, flip the display to increase in the opposite direction
<b>ScaleType</b>	One of the values <b>Linear</b> , <b>Log10</b> , and <b>SplitLinearLog10</b>
<b>Span</b>	Number of values between the minimum and maximum values

## **Angular Scale Range**

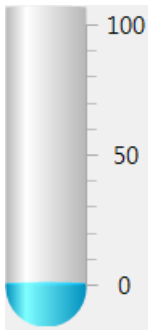
The root node of these parameters is **Instrument+ScaleRange**.

<b>Parameter</b>	<b>Usage</b>
<b>AngleMin</b>	Specify starting point of scale, from bottom of circle
<b>AngleSpan</b>	Specify number of degrees taken up by scale

# GaugeFluidLevel

Graphic instrument to display values of fluid sensor signals



## Description



Use the `GaugeFluidLevel` instrument to display real-valued data suitable for a fluid gauge, such as volume and pressure.

## Key Parameters

The key parameters are under the **Instrument** node in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

## Scale Graphic Display

The root node of this parameter is **Instrument**.

Parameter	Usage
<code>AutoSize</code>	If <code>True</code> , size the graphic to accommodate the parts of the display

The root node of these parameters is **Instrument+ScaleDisplay+GeneratorAuto**.

Parameter	Usage
<b>DesiredIncrement</b>	Display of major tick values. $\text{number of labels} = \text{span} / (\text{desired increment} + 1)$ . Does nothing if the required labels do not fit in the space available in the graphic.
<b>FixedMinMaxMajor</b>	If <b>True</b> , the top and bottom ticks are constrained to be major ticks with min/max values defined by <b>Min</b> and <b>Span</b>
<b>MidIncluded</b>	If <b>True</b> , insert a tick halfway between major ticks.  If <b>MinorCount</b> is even, space the minor ticks equally around the center tick. If <b>MinorCount</b> is odd, replace the center tick with the middle tick. If
<b>MinorCount</b>	Number of minor ticks between major ticks
<b>MinTextSpacing</b>	Minimum space between scale ticks

## Scale Text Display

The root node of these parameters is **Instrument+ScaleDisplay+TextFormatting**.

Parameter	Usage
<b>Precision</b>	Number of digits to the right of the decimal point
<b>PrecisionStyle</b>	One of the values <b>FixedDecimalPoints</b> , <b>SignificantDigits</b> , <b>None</b>
<b>Style</b>	One of the values <b>Number</b> , <b>Thousands</b> , <b>Prefix</b> , <b>Exponent</b> , <b>Price32nds</b> , <b>DateTime</b> , <b>DateTimeUTC</b>
<b>UnitsText</b>	Display unit next to tick labels



## General Scale Range

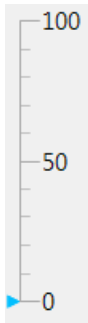
The root node of these parameters is **Instrument+ScaleRange**.

Parameter	Usage
<b>Min</b>	Minimum possible value
<b>Reverse</b>	If True, flip the display to increase in the opposite direction
<b>ScaleType</b>	One of the values <b>Linear</b> , <b>Log10</b> , and <b>SplitLinearLog10</b>
<b>Span</b>	Number of values between the minimum and maximum values

## GaugeLinear

Graphic instrument to display signal values



### Description



Use the **GaugeLinear** instrument to display real-valued data suitable for a linear gauge, such as temperature, volume, and pressure.

### Key Parameters

The key parameters are under the **Instrument** node in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

### Scale Graphic Display

The root node of this parameter is **Instrument**.

Parameter	Usage
<b>AutoSize</b>	If <b>True</b> , size the graphic to accommodate the parts of the display

The root node of these parameters is **Instrument+ScaleDisplay+GeneratorAuto**.

Parameter	Usage
<b>DesiredIncrement</b>	Display of major tick values. $\text{number of labels} = \text{span} / (\text{desired increment} + 1)$ . Does nothing if the required labels do not fit in the space available in the graphic.
<b>FixedMinMaxMajor</b>	If <b>True</b> , the top and bottom ticks are constrained to be major ticks with min/max values defined by <b>Min</b> and <b>Span</b>
<b>MidIncluded</b>	If <b>True</b> , insert a tick halfway between major ticks.  If <b>MinorCount</b> is even, space the minor ticks equally around the center tick. If <b>MinorCount</b> is odd, replace the center tick with the middle tick. If
<b>MinorCount</b>	Number of minor ticks between major ticks
<b>MinTextSpacing</b>	Minimum space between scale ticks

## Scale Text Display

The root node of these parameters is **Instrument+ScaleDisplay+TextFormatting**.

Parameter	Usage
<b>Precision</b>	Number of digits to the right of the decimal point
<b>PrecisionStyle</b>	One of the values <b>FixedDecimalPoints</b> , <b>SignificantDigits</b> , <b>None</b>
<b>Style</b>	One of the values <b>Number</b> , <b>Thousands</b> , <b>Prefix</b> , <b>Exponent</b> , <b>Price32nds</b> , <b>DateTime</b> , <b>DateTimeUTC</b>
<b>UnitsText</b>	Display unit next to tick labels

## General Scale Range

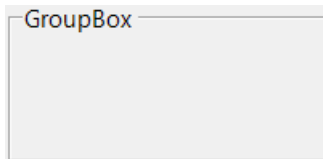
The root node of these parameters is **Instrument+ScaleRange**.

<b>Parameter</b>	<b>Usage</b>
<b>Min</b>	Minimum possible value
<b>Reverse</b>	If True, flip the display to increase in the opposite direction
<b>ScaleType</b>	One of the values <code>Linear</code> , <code>Log10</code> , and <code>SplitLinearLog10</code>
<b>Span</b>	Number of values between the minimum and maximum values

# GroupBox

Nonscrollable graphic container for instruments

## Description



The **GroupBox** graphic provides a container for other instruments. It can be stretched and shrunk at design time, but cannot be scrolled.

## Key Parameters

The key parameters are under the **Layout** node in the property list.

Parameter	Usage
<b>AutoSize</b>	If <b>True</b> , the box expands at design time to make visible the instruments within it
<b>AutoSizeMode</b>	Possible values are <b>GrowAndShrink</b> and <b>GrowOnly</b> . The default is <b>GrowOnly</b> .

# HexadecimalDisplay

Text box instrument to display signal values



## Description



The **HexadecimalDisplay** instrument displays numeric data in hexadecimal format. It is used for digital data, such as status codes and register contents.

## Key Parameters

The key parameters are under the **Instrument** node in the property list.

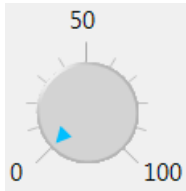
To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

Parameter	Usage
<b>AutoSize</b>	If <b>True</b> , the box expands at design time to make visible the specified digits. The default is <b>True</b> .
<b>DigitCount</b>	Number of hex digits to be displayed
<b>DigitLeading</b>	Possible values are <b>None</b> and <b>Zeros</b> .

# Knob

Graphic instrument to set parameter values



## Description



Use the **Knob** instrument to set real-valued data such as amplitude and frequency under conditions where an exact value is not required.

## Key Parameters

The key parameters are under the **Instrument** node in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

## OffSwitch Graphic Display

The root node of this parameter is **Instrument+OffSwitch**.

Parameter	Usage
Enabled	If True, the switch is visible
On	If True, the switch is on

## Scale Graphic Display

The root node of this parameter is **Instrument**.

Parameter	Usage
<b>AutoSize</b>	If <b>True</b> , size the graphic to accommodate the parts of the display

The root node of these parameters is **Instrument+ScaleDisplay+GeneratorAuto**.

Parameter	Usage
<b>DesiredIncrement</b>	Display of major tick values. $\text{number of labels} = \text{span} / (\text{desired increment} + 1)$ . Does nothing if the required labels do not fit in the space available in the graphic.
<b>FixedMinMaxMajor</b>	If <b>True</b> , the top and bottom ticks are constrained to be major ticks with min/max values defined by <b>Min</b> and <b>Span</b>
<b>MidIncluded</b>	If <b>True</b> , insert a tick halfway between major ticks.  If <b>MinorCount</b> is even, space the minor ticks equally around the center tick. If <b>MinorCount</b> is odd, replace the center tick with the middle tick. If
<b>MinorCount</b>	Number of minor ticks between major ticks
<b>MinTextSpacing</b>	Minimum space between scale ticks

### Scale Text Display

The root node of these parameters is **Instrument+ScaleDisplay+TextFormatting**.

Parameter	Usage
<b>Precision</b>	Number of digits to the right of the decimal point
<b>PrecisionStyle</b>	One of the values <b>FixedDecimalPoints</b> , <b>SignificantDigits</b> , <b>None</b>
<b>Style</b>	One of the values <b>Number</b> , <b>Thousands</b> , <b>Prefix</b> , <b>Exponent</b> , <b>Price32nds</b> , <b>DateTime</b> , <b>DateTimeUTC</b>



<b>UnitsText</b>	Display unit next to tick labels
------------------	----------------------------------

## General Scale Range

The root node of these parameters is **Instrument+ScaleRange**.

<b>Parameter</b>	<b>Usage</b>
<b>Min</b>	Minimum possible value
<b>Reverse</b>	If True, flip the display to increase in the opposite direction
<b>ScaleType</b>	One of the values <b>Linear</b> , <b>Log10</b> , and <b>SplitLinearLog10</b>
<b>Span</b>	Number of values between the minimum and maximum values

## Angular Scale Range

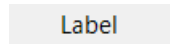
The root node of these parameters is **Instrument+ScaleRange**.

<b>Parameter</b>	<b>Usage</b>
<b>AngleMin</b>	Specify starting point of scale, from bottom of circle
<b>AngleSpan</b>	Specify number of degrees taken up by scale

# Label

Graphic container for text

## Description



Use the **Label** graphic to add text to the instrument layout.

## Key Parameters

The key parameters are under the **Appearance** and **Layout** nodes in the property list.

### Appearance Parameters

The root node of these parameters is **Appearance**.

Parameter	Usage
<b>Text</b>	Contains the text displayed by the label
<b>TextAlign</b>	<p>Specifies left-right, top-bottom alignment using a 3x3 matrix.</p> <p>This display represents setting TopLeft.</p>

### Layout Parameters

The root node of this parameter is **Layout**.

---

Parameter	Usage
AutoSize	If True, size the graphic to accommodate the text

## LED

Graphic instrument to display signal values



### Description



Use the **LED** instrument to display binary (1 or 0) data.

### Key Parameters

The key parameters are under the **Instrument** node in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

### General Parameters

The root node of these parameters is **Instrument**.

Parameter	Usage
<b>AutoSize</b>	If <b>True</b> , size the graphic to accommodate the specified graphic parameters.
<b>BlinkerEnable</b>	If <b>True</b> , <b>LED</b> graphic blinks continuously.

### Indicator Parameters

The root node of these parameters is **Instrument+Indicator**.

Parameter	Usage
<b>ColorActive</b>	Indicator color if signal value is 1.

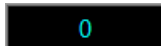
---

Parameter	Usage
<b>ColorInactive</b>	Indicator color if signal value is 0.

# NumericDisplay

Text box instrument to display signal values



## Description



Use the **NumericDisplay** instrument to display real-valued data in specified formats.

## Key Parameters

The key parameters are under the **Instrument** and **Iocomp** nodes in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

## General Parameters

The root node of this parameter is **Instrument**.

Parameter	Usage
<b>AutoSize</b>	If <b>True</b> , the box expands at design time to make visible the specified digits. The default is <b>True</b> .

## Value Display

The root node of these parameters is **Iocomp+TextFormatting**.

Parameter	Usage
<b>Precision</b>	Number of digits to the right of the decimal point

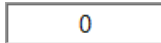
---

<b>PrecisionStyle</b>	One of the values <code>FixedDecimalPoints</code> , <code>SignificantDigits</code> , <code>None</code>
<b>Style</b>	One of the values <code>Number</code> , <code>Thousands</code> , <code>Prefix</code> , <code>Exponent</code> , <code>Price32nds</code> , <code>DateTime</code> , <code>DateTimeUTC</code>
<b>UnitsText</b>	Display unit next to tick labels

# NumericEntry

Text box instrument to set parameter values



## Description



Use the **NumericEntry** instrument to enter real-valued data in specified formats under conditions where an exact value is required.

## Key Parameters

The key parameters are under the **Instrument** node in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

## Text Display

The root node of these parameters is **Instrument+TextFormatting**.

Parameter	Usage
<b>Precision</b>	Number of digits to the right of the decimal point
<b>PrecisionStyle</b>	One of the values <code>FixedDecimalPoints</code> , <code>SignificantDigits</code> , <code>None</code>
<b>Style</b>	One of the values <code>Number</code> , <code>Thousands</code> , <code>Prefix</code> , <code>Exponent</code> , <code>Price32nds</code> , <code>DateTime</code> , <code>DateTimeUTC</code>
<b>UnitsText</b>	Display unit next to tick labels



# NumericUpDownEntry

Text box instrument to set parameter values



## Description



Use the **NumericUpDownEntry** instrument to enter real-valued data and increment it by a specified amount under conditions where a step change is required.

## Key Parameters

The key parameters are under the **Layout** and **Data** nodes in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

## General Parameters

The root node of this parameter is **Layout**.

Parameter	Usage
AutoSize	If True, the box expands at design time to make visible the specified digits. The default is False.

## Scale Range

The root node of these parameters is **Data**.

Parameter	Usage
-----------	-------

<b>DecimalPlaces</b>	Number of decimal places to display
<b>Increment</b>	Value to add or subtract in response to an up-arrow or down-arrow
<b>Maximum</b>	Maximum data value
<b>Minimum</b>	Minimum data value

# Panel

Scrollable graphic container for instruments

## Description



The **Panel** graphic provides a container for other instruments. You can stretch and shrink it at design time and scroll it at run time.

## Key Parameters

The key parameters are under the **Layout** node in the property list.

Parameter	Usage
<b>AutoScroll</b>	If <b>True</b> , the box scrolls at run time to make fully visible partially visible instruments within it.
<b>AutoSize</b>	If <b>True</b> , the box expands at design time to make visible the instruments within it.
<b>AutoSizeMode</b>	Possible values are <b>GrowAndShrink</b> and <b>GrowOnly</b> . The default is <b>GrowOnly</b>

## PictureBox

Graphic container for pictures



### Description



The **PictureBox** graphic provides a container for graphics, for example a photograph or line drawing.

### Key Parameters

The key parameter is under the **Behavior** node in the property list.

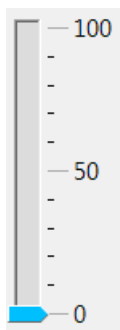
To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

Parameter	Usage
<b>SizeMode</b>	Possible values are Normal, StretchImage, AutoSize, CenterImage, and Zoom. The default is Normal

# Slider

Graphic instrument to set parameter values



## Description



Use the **Slider** instrument to set real-valued data such as temperature and pressure under conditions where the exact value is not required.

## Key Parameters

The key parameters are under the **Instrument** node in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

## Scale Graphic Display

The root node of this parameter is **Instrument**.

Parameter	Usage
AutoSize	If True, size the graphic to accommodate the parts of the display

The root node of these parameters is **Instrument+ScaleDisplay+GeneratorAuto**.

Parameter	Usage
<b>DesiredIncrement</b>	Display of major tick values. $\text{number of labels} = \text{span} / (\text{desired increment} + 1)$ . Does nothing if the required labels do not fit in the space available in the graphic.
<b>FixedMinMaxMajor</b>	If <b>True</b> , the top and bottom ticks are constrained to be major ticks with min/max values defined by <b>Min</b> and <b>Span</b>
<b>MidIncluded</b>	If <b>True</b> , insert a tick halfway between major ticks.  If <b>MinorCount</b> is even, space the minor ticks equally around the center tick. If <b>MinorCount</b> is odd, replace the center tick with the middle tick. If
<b>MinorCount</b>	Number of minor ticks between major ticks
<b>MinTextSpacing</b>	Minimum space between scale ticks

## Scale Text Display

The root node of these parameters is **Instrument+ScaleDisplay+TextFormatting**.

Parameter	Usage
<b>Precision</b>	Number of digits to the right of the decimal point
<b>PrecisionStyle</b>	One of the values <b>FixedDecimalPoints</b> , <b>SignificantDigits</b> , <b>None</b>
<b>Style</b>	One of the values <b>Number</b> , <b>Thousands</b> , <b>Prefix</b> , <b>Exponent</b> , <b>Price32nds</b> , <b>DateTime</b> , <b>DateTimeUTC</b>
<b>UnitsText</b>	Display unit next to tick labels

## General Scale Range

The root node of these parameters is **Instrument+ScaleRange**.

Parameter	Usage
<b>Min</b>	Minimum possible value
<b>Reverse</b>	If <b>True</b> , flip the display to increase in the opposite direction
<b>ScaleType</b>	One of the values <b>Linear</b> , <b>Log10</b> , and <b>SplitLinearLog10</b>
<b>Span</b>	Number of values between the minimum and maximum values

## SwitchLED

Graphic instrument to set parameter values



### Description



Use the **SwitchLED** instrument to set a binary (1 or 0) value.

### Key Parameters

The key parameters are under the **Instrument** node in the property list.

To access a parameter dialog box for the instrument as a whole, select the instrument and click the Tasks button  in the top right corner. To access a dialog box for a parameter group, click the group, and then click the continuation button  to the right of the group.

### General Parameters

The root node of these parameters is **Instrument**.

Parameter	Usage
AutoSize	If <b>True</b> , size the graphic to accommodate the specified graphic parameters.
Text	Receives visible text on switch.

### Indicator Parameters

The root node of these parameters is **Instrument+Indicator**.

Parameter	Usage
ColorActive	Indicator color if signal value is 1.



Parameter	Usage
ColorInactive	Indicator color if signal value is 0.



# Target Computer Command-Line Interface Reference

---

## Target Computer Commands

You have a limited set of commands that you can use to work the real-time application after it has been loaded to the target computer, and to interface with the scopes for that application.

The target computer command-line interface enables you to work with target and scope objects in a limited capacity. Functions let you interact directly with the scope or target. Property commands let you work with target and scope properties. Variable commands let you alias target computer command-line interface commands to names of your choice.

Refer to “Control Real-Time Application at Target Computer Command Line” for a description of how to use these functions and commands.

### In this section...

“Target Object Function Commands” on page 5-2

“Target Object Property Commands” on page 5-3

“Scope and Video Object Function Commands” on page 5-4

“Scope Object Property Commands” on page 5-6

“Aliasing with Variable Commands” on page 5-10

## Target Object Function Commands

When you are using the target computer command-line interface, target object functions are limited to starting and stopping the real-time application.

The following table lists the syntax for the target commands that you can use on the target computer. The equivalent MATLAB syntax is shown in the right column. The target object name `tg` is used as an example for the MATLAB functions. These functions assume that you have already loaded the real-time application onto the target computer.

Target Computer Command	Description	MATLAB Equivalent
<code>start</code>	Start the real-time application currently loaded on the target computer.	<code>start(tg)</code>

Target Computer Command	Description	MATLAB Equivalent
stop	Stop the real-time application currently running on the target computer.	stop(tg)
reboot	Restart the target computer.	reboot(tg)

## Target Object Property Commands

When you are using the target computer command-line interface, target object properties are limited to parameters, signals, stop time, and sample time. Note the difference between a parameter index (0, 1, . . .) and a parameter name (P0, P1, . . .).

The following table lists the syntax for the target commands that you can use to manipulate target object properties. The MATLAB equivalent syntax is shown in the right column, and the target object name `tg` is used as an example for the MATLAB functions.

Target Computer Command	Description	MATLAB Equivalent
getpar param_index	Display the value of a block parameter using the parameter index.	getparam(tg, param_index)
setpar param_index = number	Change the value of a block parameter using the parameter index.	setparam(tg, param_index, number)
stoptime = number	With the value <code>number</code> , run for the specified number of seconds.	tg.StopTime = number
stoptime = Inf	With the value <code>Inf</code> , run the real-time application until you manually stop it or reset the target computer.	tg.StopTime = Inf
sampletime = number	Enter a new sample time.	tg.SampleTime = number
P#	Display the value of the block parameter with index #.	getparam(tg, param_index)

Target Computer Command	Description	MATLAB Equivalent
	For example, P2 displays the value of block parameter 2.	
S#	Display the value of the signal with index #.  For example, S2 displays the value of signal 2.	getsignal(tg, sig_index)

## Scope and Video Object Function Commands

When using the target computer command-line interface, you use scope object functions to start a scope and add signal traces. You can also collapse scopes and video displays into icons and expand them again. Notice that the functions `addscope` and `remscope` are target object functions on the development computer, and notice the difference between a signal index (0, 1, . . .) and a signal name (S0, S1, . . .).

The following table lists the syntax for the target commands that you can use on the target computer. The MATLAB equivalent syntax is shown in the right column. The target object name `tg` and the scope object name `sc` are used as an example for the MATLAB functions.

Target Computer Command	Description	MATLAB Equivalent
<code>addscope</code>	Without an argument, add a target scope and assign it the next available index.	<code>addscope(tg, 'target')</code>
<code>addscope scope_index</code>	With argument <code>scope_index</code> , add a target scope and assign it index <code>scope_index</code> .	<code>addscope(tg, 'target', scope_index)</code>
<code>remscope scope_index</code>	With value <code>scope_index</code> , remove scope <code>scope_index</code> .	<code>remscope(tg, scope_index)</code>
<code>remscope all</code>	With value <code>all</code> , remove all scopes.	<code>remscope(tg)</code>

Target Computer Command	Description	MATLAB Equivalent
startscope scope_index  startscope all	With value <code>scope_index</code> , start the scope with index <code>scope_index</code> .  With value <code>all</code> , start all scopes.	<code>start(sc)</code>  <code>start(getscope(tg))</code>
stopscope scope_index  stopscope all	With value <code>scope_index</code> , stop the scope with index <code>scope_index</code> .  With value <code>all</code> , stop all scopes.	<code>stop(sc)</code>  <code>stop(getscope(tg))</code>
addsignal scope_index = sig_index1, sig_index2, ...	With values <code>sig_index1</code> , <code>sig_index2</code> , ..., add the signals with these signal indexes to the scope with index <code>scope_index</code> .	<code>addsignal(sc, sig_index_vector)</code>
remsignal scope_index = sig_index1, sig_index2, ...  remsignal scope_index	With values <code>sig_index1</code> , <code>sig_index2</code> , ..., remove the signals with these signal indexes from the scope with index <code>scope_index</code> .  Without a <code>sig_index</code> value, remove all the signals from the scope with index <code>scope_index</code> .	<code>remsignal(sc, sig_index_vector)</code>  <code>remsignal(sc)</code>
show Scope scope_index	With value <code>scope_index</code> , expand scope <code>scope_index</code> from an icon.	
hide Scope scope_index	With value <code>scope_index</code> , collapse scope <code>scope_index</code> into an icon.	
show Video video_index	With value <code>video_index</code> , expand video display <code>video_index</code> from an icon.	

Target Computer Command	Description	MATLAB Equivalent
hide Video video_index	With value video_index, collapse video display video_index into an icon.	

## Scope Object Property Commands

When you use the target computer command-line interface, scope object properties are limited to those shown in the following table. Notice the difference between a scope index (0, 1, . . .) and the MATLAB variable name for the scope object on the development computer. The scope index is indicated in the top left corner of a scope window (SC0, SC1, . . .).

If a scope is running, you need to stop the scope before you can change a scope property.

The following table lists the syntax for the target properties that you can set on the target computer. The equivalent MATLAB syntax is shown in the right column. The scope object name SC is used as an example for the MATLAB functions

Target Computer Command	Description	MATLAB Equivalent
numsamples scope_index = number	Set the number of contiguous samples captured by scope scope_index to number.	sc.NumSamples = number
decimation scope_index = 1	With value 1, the scope returns all sample points.	sc.Decimation = 1
decimation scope_index = number	With value n, the scope returns every nth sample point.	sc.Decimation = number
grid scope_index on grid scope_index off	With value on, the scope grid display is visible.  With value off, the scope grid display is not visible.	sc.Grid = 'on'  sc.Grid = 'off'
scopemode scope_index = 0	With value 0 or numerical, scope scope_index	sc.DisplayMode = 'numerical'



Target Computer Command	Description	MATLAB Equivalent
<code>scopemode scope_index = numerical</code>	displays signal values as text.	<code>sc.DisplayMode = 'redraw'</code>
<code>scopemode scope_index = 1</code>	With value 1 or <code>redraw</code> , scope <code>scope_index</code> plots signal values when <code>numsamples</code> samples has been acquired.	<code>sc.DisplayMode = 'rolling'</code>
<code>scopemode scope_index = redraw</code>		
<code>scopemode scope_index = 3</code>	With value 3 or <code>rolling</code> , scope <code>scope_index</code> plots signal values at every sample time.	
<code>scopemode scope_index = rolling</code>		
	<b>Note:</b> Value 2, <code>sliding</code> , will be removed in a future release. It behaves like value 3, <code>rolling</code> .	

Target Computer Command	Description	MATLAB Equivalent
triggermode scope_index = 0	With value 0 or freerun, scope scope_index triggers on every sample time.	sc.TriggerMode = 'freerun'
triggermode scope_index = freerun	With value 1 or software, scope scope_index triggers from Command Window.	sc.TriggerMode = 'software'
triggermode scope_index = 1	With value 2 or signal, scope scope_index triggers when a designated signal changes state.	sc.TriggerMode = 'signal'
triggermode scope_index = software	With value 3 or scope, scope scope_index triggers when a designated scope triggers.	sc.TriggerMode = 'scope'
triggermode scope_index = 2		
triggermode scope_index = signal		
triggermode scope_index = 3		
triggermode scope_index = scope		
numprepostsamples scope_index = number	Number of samples collected before or after a trigger event.	sc.NumPrePostSamples = number
triggersignal scope_index = sig_index	If triggermode is signal, triggersignal identifies the block output signal to use for triggering the scope.	sc.TriggerSignal = sig_index
triggersample scope_index = number	If triggermode is scope, triggersample specifies which sample of the triggering scope the current scope triggers on.	sc.TriggerSample = number

Target Computer Command	Description	MATLAB Equivalent
triggerlevel scope_index = number	If triggermode is signal, triggerlevel indicates the value the signal has to cross to trigger the scope to start acquiring data.	sc.TriggerLevel = number
triggerslope scope_index = 0	If triggermode is signal:  With value 0 or either, the signal triggers the scope when it crosses triggerlevel in either the rising or falling directions.  With value 1 or rising, the signal triggers the scope when it crosses triggerlevel in the rising direction.  With value 2 or falling, the signal triggers the scope when it crosses triggerlevel in the falling direction.	sc.TriggerSlope = 'Either'
triggerslope scope_index = either		sc.TriggerSlope = 'Rising'
triggerslope scope_index = 1		sc.TriggerSlope = 'Falling'
triggerslope scope_index = rising		
triggerslope scope_index = 2		
triggerslope scope_index = falling		
triggerscope scope_index = scope_index2	If triggermode is scope, triggerscope identifies the scope to use for a trigger.	sc.TriggerScope = scope_index2
triggerscopesample scope_index= integer	If triggermode is scope, triggerscopesample specifies which sample of the triggering scope to trigger on.	sc.TriggerScopeSample = integer

Target Computer Command	Description	MATLAB Equivalent
<code>ylim limit scope_index = min_y, max_y</code>	With value <code>min_y</code> , <code>max_y</code> , change the lower and upper <i>y</i> -axis values to <code>min_y</code> and <code>max_y</code> .	<code>sc.YLimit = [min_y, max_y]</code>
<code>ylim limit scope_index = auto</code>	With value <code>auto</code> , allow the lower and upper <i>y</i> -axis values to be determined by the values being displayed.	<code>sc.YLimit = 'auto'</code>

## Aliasing with Variable Commands

You can set a variable to a command string, and later use that variable to execute that command. For example, type the following on the target computer command line:

```
setvar aa = startscope 2
setvar bb = stopscope 2
```

Later, to start and stop scope 2, you can type the following:

```
aa
bb
```

The following table lists the syntax for the aliasing variable commands that you can use only on the target computer. There is no MATLAB equivalent syntax. For a usage example, see “Alias Commands at Target Computer Command Line”.

Target Computer Command	Description
<code>setvar variable_name = command</code>	Set a variable to a target computer command line string.
<code>getvar variable_name</code>	Display the value of a variable.
<code>delvar variable_name</code>	Delete a variable.
<code>delallvar</code>	Delete all variables.
<code>showvar</code>	Display a list of variables.

# Simulink Real-Time Performance Advisor Checks

---

## Simulink Real-Time Performance Advisor Checks

### In this section...

“Performance Advisor Check Overview” on page 6-2

“Baseline” on page 6-2

“System Target File Compatibility” on page 6-3

“Profiling Settings” on page 6-3

“Check Target” on page 6-3

“Real-Time Performance Baseline” on page 6-4

“Determine minimum sample time” on page 6-4

“Real-Time” on page 6-5

“Outport Logging” on page 6-5

“EtherCAT Synchronous SDO” on page 6-5

“Concurrent execution” on page 6-6

“Final Validation” on page 6-6

### Performance Advisor Check Overview

Use Performance Advisor checks to improve real-time application execution time. Performance Advisor runs the check and only provides recommendations. It does not modify your model.

#### See Also

“Improve Simulation Performance Using Performance Advisor”

### Baseline

Checks preconditions for real-time execution, and then performs an initial measurement to collect baseline performance data. Performance Advisor later uses this baseline to compare performance before and after you implement the improvements that Performance Advisor recommends.

#### See Also

- “Performance Optimization”

- “Improve Simulation Performance Using Performance Advisor ”

## System Target File Compatibility

Checks that the system target file is compatible with the real-time advisor workflow.

In the Configuration Parameters **Code Generation** pane, set the **System target file** setting to either `slrt.tlc` or `slrtert.tlc`.

### See Also

- “Set Configuration Parameters”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor ”

## Profiling Settings

Checks that the execution profiling settings are compatible with the real-time advisor workflow.

In the Configuration Parameters **Verification** pane, select the **Measure task execution time** check box.

### See Also

- “Configure Real-Time Application for Profiling”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor ”

## Check Target

Pings the target computer and verifies that it is in a clean state.

To clear the target computer of prior simulations, select the **Automatically unload any loaded application** check box.

### See Also

- “Performance Optimization”

- “Improve Simulation Performance Using Performance Advisor ”

### Real-Time Performance Baseline

Collects execution-time measurements and establishes a performance baseline.

This check builds, downloads, and executes the real-time application on the target computer. When the check passes, it displays the following information:

- **Margin before CPU overload (0% indicates CPU overload)** — A table containing the real-time application task name, the sample rate, and the margin.

Margin is the minimum value of headroom for a task over all the measured samples.

Headroom is the time between the end of execution and the start of the next sample, as a percentage of sample time. For example, if the sample time is 1.2 ms and a task takes 0.93 ms to execute, the headroom is  $(1.2 - 0.93) / 1.2$ , or 22.5%.

As the margin approaches 0%, the application gets closer to overloading the CPU.

- **Average CPU Usage** — A pie chart showing the average CPU resources that the real-time application uses, as a percentage of available resources.

The available CPU resources include all of the processors on a multicore target computer. For example, a single-tasking model running on a quad-core processor cannot exceed 25% CPU usage.

The background task aggregates the CPU time for all operating system tasks that are not related to application execution. These tasks include updating the target screen, communicating with the development computer, and so on. It also includes the time that the CPU is idle.

#### See Also

- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor ”

### Determine minimum sample time

To determine the minimum sample time possible for the model, this check iteratively runs the model, decreasing the sample time on each run. The algorithm stops at a sample time that is greater than a sample time that can cause an overload.



Random factors such as network latency can change the minimum sample time of a model. As a best practice, set the sample time for your model to a value greater than the minimum sample time returned by this check.

This check builds, downloads, and executes the real-time application.

### **See Also**

- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor ”

## **Real-Time**

Checks the real-time application and application setup, and recommends changes to improve performance.

### **See Also**

- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor ”

## **Output Logging**

Checks for the use of **Output** blocks for data logging. Data logging using **Output** blocks can slow down execution.

As an alternative, consider using a real-time **Scope** block configured as a file scope.

### **See Also**

- **Scope**
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor ”

## **EtherCAT Synchronous SDO**

Checks for the use of **EtherCAT Synchronous SDO** blocks. **EtherCAT Synchronous SDO** blocks can slow down execution.

As an alternative, consider using EtherCAT Asynchronous SDO blocks.

### See Also

- “EtherCAT”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor ”

### Concurrent execution

Checks if you can perform concurrent execution of the real-time application on a multicore target computer.

To perform concurrent execution:

- Choose a multicore target computer.
- In the Simulink Real-Time Explorer **Target settings** pane, select the **Multicore CPU** check box.
- In the Configuration Parameters **Solver** pane, select the **Allow tasks to execute concurrently on target** check box.

This check updates the model diagram.

### See Also

- “Concurrent Execution Using Multicore Target Computer”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor ”

### Final Validation

Validates the overall performance improvement that your changes make in real-time execution time and accuracy.

If you have not validated the performance improvement resulting from other checks, use this check to perform a final validation of the changes to the model.

This check builds, downloads, and executes the real-time application. When the check passes, it displays the following information for the baseline and final validation runs:

- **Margin before CPU overload (0% indicates CPU overload)** — A table containing, for each run, the real-time application task name, the sample rate, and the margin.

Margin is the minimum value of headroom for a task over all the measured samples.

Headroom is the time between the end of execution and the start of the next sample, as a percentage of sample time. For example, if the sample time is 1.2 ms and a task takes 0.93 ms to execute, the headroom is  $(1.2 - 0.93) / 1.2$ , or 22.5%.

As the margin approaches 0%, the application gets closer to overloading the CPU.

- **Average CPU Usage** — Pie charts showing, for each run, the average CPU resources that the real-time application uses, as a percentage of available resources.

The available CPU resources include all of the processors on a multicore target computer. For example, a single-tasking model running on a quad-core processor cannot exceed 25% CPU usage.

The background task aggregates the CPU time for all operating system tasks that are not related to application execution. These tasks include updating the target screen, communicating with the development computer, and so on. It also includes the time that the CPU is idle.

### See Also

- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor ”
- “Comparing Performance”

