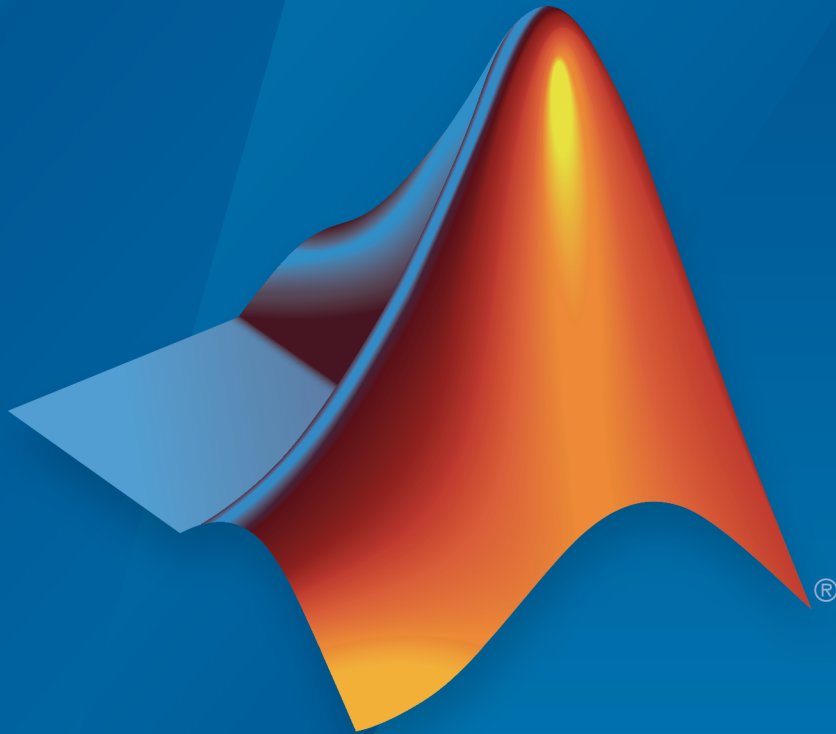


Simulink[®] Real-Time[™]

Reference



MATLAB[®]&SIMULINK[®]

R2017a



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



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

Simulink[®] Real-Time[™] Reference

© COPYRIGHT 2002–2017 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 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 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)
September 2016	Online only	Revised for Version 6.5 (Release 2016b)
March 2017	Online only	Revised for Version 6.6 (Release 2017a)

Configuration Parameters

Simulink Real-Time Options Pane	1-2
Configuration	1-3
Tips	1-4
To get help on an option	1-4
Build for default target computer	1-5
Settings	1-5
Dependency	1-5
Command-Line Information	1-5
See Also	1-5
Specify target computer name	1-6
Settings	1-6
Tip	1-6
Dependencies	1-6
Command-Line Information	1-6
See Also	1-6
Automatically download application after building	1-7
Settings	1-7
Command-Line Information	1-7
See Also	1-7
Name of Simulink Real-Time object created by build process	1-8
Settings	1-8
Tip	1-8
Command-Line Information	1-8
See Also	1-8
Use default communication timeout	1-9
Settings	1-9

Dependencies	1-9
Command-Line Information	1-9
See Also	1-9
Specify the communication timeout in seconds	1-10
Settings	1-10
Tip	1-10
Dependencies	1-10
Command-Line Information	1-10
See Also	1-10
Execution mode	1-11
Settings	1-11
Command-Line Information	1-11
Real-time interrupt source	1-12
Settings	1-12
Tips	1-12
Command-Line Information	1-12
I/O board generating the interrupt	1-13
Settings	1-13
Command-Line Information	1-14
PCI slot (-1: autosearch) or ISA base address	1-16
Settings	1-16
Tip	1-16
Command-Line Information	1-16
See Also	1-16
Log Task Execution Time	1-17
Settings	1-17
Command-Line Information	1-17
See Also	1-17
Signal logging data buffer size in doubles	1-18
Settings	1-18
Tips	1-18
Command-Line Information	1-18
Number of profiling events (each uses 20 bytes)	1-20
Settings	1-20
Tips	1-20

Command-Line Information	1-20
See Also	1-20
Double buffer parameter changes	1-21
Settings	1-21
Tips	1-21
Command-Line Information	1-21
Load a parameter set from a file on the designated target file	
system	1-22
Settings	1-22
Dependencies	1-22
Command-Line Information	1-22
See Also	1-22
File name	1-23
Settings	1-23
Tip	1-23
Dependencies	1-23
Command-Line Information	1-23
Generate INCA/CANape extensions (disables Simulation Data	
Inspector and Dashboard blocks)	1-24
Settings	1-24
Command-Line Information	1-24
See Also	1-24
Enable Stateflow animation	1-25
Settings	1-25
Command-Line Information	1-25
See Also	1-25

2

3

4

Instrumentation for Real-Time Applications	4-2
Instrument Selection and Binding	4-4
Display Instruments	4-5
Tuning Instruments	4-7
Layout Elements	4-8
Graphical Properties	4-10
Explorer Configuration Exported to Run Outside MATLAB	4-16
Execution Environment	4-16
Signal Groups	4-16
Parameter Groups	4-17
Instrument Panels	4-17
Window Layout	4-17
Limitations as Standalone Executable	4-17
View Signal Waveforms with Scope Instrument	4-19
Create Instrument Panel	4-20
Configure Slider Instrument for Parameter Tuning	4-20
Configure Scope Instrument with Default Triggering	4-22
Run Instrumented Model	4-23
View Signal Waveforms with Signal Triggered Scope Instrument	4-27
Preconditions	4-27
Configure Scope Instrument	4-27
Run Instrumented Model	4-28

Instrument a Tank Model	4-32
Save Environment Properties	4-40
Save and Load Instrument Panels	4-41
Save and Restore Layouts	4-42
Prepare Explorer Environment for Export	4-43
Prepare Instrument Panel Configuration for Export	4-45
Export Explorer Configuration	4-48
Unpack and Run Standalone Configuration	4-49

Simulink Real-Time Explorer Instruments

5

Target Computer Command-Line Interface Reference

6

Target Computer Commands	6-2
Target Object Function Commands	6-2
Target Object Property Commands	6-3
Scope and Video Object Function Commands	6-4
Scope Object Property Commands	6-6
Aliasing with Variable Commands	6-10

Simulink Real-Time Performance Advisor Checks

7

Simulink Real-Time Performance Advisor Checks	7-2
To get help on an option	7-3
See Also	7-3
Baseline	7-4
See Also	7-4
System Target File Compatibility	7-5
See Also	7-5
Profiling Settings	7-6
See Also	7-6
Check Target	7-7
See Also	7-7
Real-Time Performance Baseline	7-8
See Also	7-8
Determine minimum sample time	7-9
See Also	7-9
Real-Time	7-10
See Also	7-10
Outport Logging	7-11
See Also	7-11
EtherCAT Synchronous SDO	7-12
See Also	7-12
Concurrent execution	7-13
See Also	7-13
Final Validation	7-14
See Also	7-14

Configuration Parameters

Simulink Real-Time Options Pane

Parameter	Description
“Build for default target computer” on page 1-5	Direct Simulink Coder™ to download the real-time application to the default target computer.
“Specify target computer name” on page 1-6	Specify a target computer name for your real-time application.
“Automatically download application after building” on page 1-7	Enable Simulink Coder to build and download the real-time application to the target computer.
“Name of Simulink Real-Time object created by build process” on page 1-8	Enter the name of the target object created by the build process.
“Use default communication timeout” on page 1-9	Direct Simulink Real-Time software to wait 5 (default) seconds for the real-time application to be downloaded to the target computer.
“Specify the communication timeout in seconds” on page 1-10	Specify a timeout, in seconds, for an attempted communication with the target computer.
“Execution mode” on page 1-11	Specify execution mode of downloaded code.
“Real-time interrupt source” on page 1-12	Select a real-time interrupt source from the I/O board.
“I/O board generating the interrupt” on page 1-13	Specify the board interrupt source.
“PCI slot (-1: autosearch) or ISA base address” on page 1-16	Enter the slot number or base address for the I/O board generating the interrupt.
“Log Task Execution Time” on page 1-17	Log task execution times to the target object property <code>tg.TETlog</code> .
“Signal logging data buffer size in doubles” on page 1-18	Enter the maximum number of sample points that the software stores before wrapping.
“Number of profiling events (each uses 20 bytes)” on page 1-20	Enter the maximum of events that the software logs for the profiling tool.

Parameter	Description
“Double buffer parameter changes” on page 1-21	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.
“Load a parameter set from a file on the designated target file system” on page 1-22	Automatically load a parameter set from a file on the designated target computer file system.
“File name” on page 1-23	Specify the target computer file name from which to load the parameter set.
“Generate INCA/CANape extensions (disables Simulation Data Inspector and Dashboard blocks)” on page 1-24	Enable real-time applications to generate data, such as A2L data, for Vector CANape [®] and ETAS [®] Inca.
“Enable Stateflow animation” on page 1-25	Enables visualization of Stateflow [®] chart animation.

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 `slrt.tlc`. This setting generates system target code for Simulink Real-Time.

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`. To retain the updated setting, save the updated model.

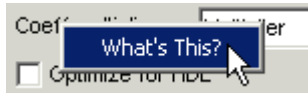
- 2 Select **C** for the **Language** parameter on the code generation pane.

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.

To get help on an option

- 1 Right-click the option text label.
- 2 From the context menu, select **What's This**.



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: character vector

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: character vector

Value: Any valid target computer

Default: ''

See Also

Simulink Real-Time Explorer

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: character vector

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: RL320objectName

Type: character vector

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

On

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

Off

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: character vector

Value: 'on' | 'off'

Default: 'on'

See Also

“Communications Timeout”

Specify the communication timeout in seconds

Specify a timeout, in seconds, for an attempted communication with 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: character vector

Value: Any valid number of seconds

Default: '5'

See Also

“Communications Timeout”

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, clear the check box for **Treat each discrete rate as a separate task**.

Command-Line Information

Parameter: RL32ModeModifier

Type: character vector

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 plugin boards like ISA boards. For PCI parallel port plugin boards, set the IRQ manually.
- Multiple boards can share an interrupt number.

Command-Line Information

Parameter: RL32IRQSourceModifier

Type: character vector

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

Bitflow NEON

Specifies that the interrupt source is the BitFlow™ NEON video 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® 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.
Scramnet_SC150+

Specifies that the interrupt source is the Systran[®] Scramnet+ SC150 board.
Softing_CAN-AC2-104

Specifies that the interrupt source is the Softing[®] CAN-AC2-104 board.
Softing_CAN-AC2-PCI

Specifies that the interrupt source is the Softing CAN-AC2-PCI board.
Speedgoat_IO321

Specifies that the interrupt source is the Speedgoat IO321 FPGA board.
Speedgoat_IO331

Specifies that the interrupt source is the Speedgoat IO331 FPGA board.
Speedgoat_IO333

Specifies that the interrupt source is the Speedgoat IO333 FPGA board.
UEI_MF_x

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: character vector

Value: 'ATI-RP-R5' |
'Bitflow NEON' |
'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' |
'Scramnet_SC150+' |
'Softing_CAN-AC2-104' |
'Softing_CAN-AC2-PCI' |
'Speedgoat_IO321' |
'Speedgoat_IO331' |
'Speedgoat_IO333' |
'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: character vector

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`.

Task execution time (TET) measures how long it takes the kernel to run for one base-rate time step. For a multirate model, use the profiler to find out what the execution time is for each rate.

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: character vector

Value: `'on'` | `'off'`

Default: `'on'`

See Also

“Signal Logging Basics”

Signal logging data buffer size in doubles

Enter the maximum number of sample points that the software stores 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 $\text{floor}(100000 / 6) = 16666$. 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: character vector

Value: '100000' | any valid memory size

Default: '100000'

Number of profiling events (each uses 20 bytes)

Enter the maximum of events that the software logs for the profiling tool.

Settings

Default: 5000

The maximum number of events that the software logs for the profiling tool.

Tips

- An event is the start of end of an interrupt or iteration of the model. For example, one sample can 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 timestamp readings. Each event occupies 20 bytes.

Command-Line Information

Parameter: xPCRL32EventNumber

Type: character vector

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 MATLAB variables as the value of block parameters, the software ignores this double buffering setting. Normal parameter tuning occurs.

Command-Line Information

Parameter: xpcDb1Buff

Type: character vector

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: character vector

Value: 'on' | 'off'

Default: 'off'

See Also

“Save and Reload Parameters with 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: character vector

Value: Any valid file name

Default: ''

Generate INCA/CANape extensions (disables Simulation Data Inspector and Dashboard blocks)

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: GenerateASAP2

Type: character vector

Value: 'on' | 'off'

Default: 'off'

See Also

“Prepare ASAP2 Data Description File”

Enable Stateflow animation

Enables visualization of Stateflow 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: character vector

Value: 'on' | 'off'

Default: 'off'

See Also

“Animate Stateflow Charts with 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', '-option_name1=option_value1 -option_nameN=option_valueN')`.

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

```
set_param(model_name, 'TLCOptions', '-axPCMaxOverloads=20  
-axPCModelStackSizeKB=4096)
```

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

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

```
ans =
```

```
-axPCMaxOverloads=20 -axPCModelStackSizeKB=4096
```

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

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

Target Computer Overload

xPCMaxOverloads — Number of acceptable CPU overloads in a real-time application execution

0 (default) | scalar

When `xPCMaxOverloads` is set to a value, the Simulink Real-Time software stops execution with a CPU overload at the next overload within the same application

execution. For example, if `xPCMaxOverloads` is set to 3, the software stops with a CPU overload at the fourth overload in the same application execution.

The default value of 0 means that overloads are registered on the first step.

Allowing the target computer CPU to overload can cause incorrect results, especially for multirate models. Use these options only for diagnosis. When your diagnosis is complete, turn off these options.

Example: `-axPCMaxOverloads=3`

xPCMaxOverloadLen — Number of acceptable overloads, in units of sample time, within the same execution step

0 (default) | scalar

When `xPCMaxOverloadLen` is set to a value, the software stops execution with a CPU overload at the next overload within the same execution step. For example, if `xPCMaxOverloadLen` is set to 2, the software stops execution with a CPU overload at the third overload within the same execution step.

The default value of 0 means that overloads are registered on the first step.

Specify a value that is less than or equal to the value for `xPCMaxOverloads`. If `xPCMaxOverload` is set to a value, for example 4, and `xPCMaxOverloadLen` is not defined, the real-time application stops if one of following occurs:

- The cumulative overloads since execution start is greater than 4.
- One execution step has two overloads.

Allowing the target computer CPU to overload can cause incorrect results, especially for multirate models. Use these options only for diagnosis. When your diagnosis is complete, turn off these options.

Example: `-axPCMaxOverloadLen=2`

Dependency

To use `xPCMaxOverloadLen`, you must first set `xPCMaxOverloads` to a value greater than or equal to `xPCMaxOverloadLen`.

xPCStartupFlag — Number of executions of the model at start up

1 (default) | scalar

`xPCStartupFlag` temporarily disables 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.

The default value of 1 means that overloads are ignored on the first step. If `xPCMaxOverloads` and `xPCMaxOverloadLen` are not set, their default setting governs.

Example: `-axPCStartupFlag=3`

Target Computer Memory

xPCModelStackSizeKB — Size of stack memory on the target computer, in kilobytes
2048 (default) | scalar

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

Target computer memory for the real-time application executable, the kernel, and other uses is limited to a maximum of 4 GB.

Example: `-axPCModelStackSizeKB=4096`

Polling Mode

xpcCPULockPoll — Turns on polling mode
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 target computer CPU clock rate. If the value is 0 or if the option is not defined, the kernel executes in interrupt mode.

Example: `-axpcCPULockPoll=1`

Floating-Point Processing

SLRFTZOFF — 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 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`

See Also

Topics

“CPU Overload Options”

“Polling Mode”

External Websites

www.ieee.org

www.microsoft.com

Simulink Real-Time Explorer

Simulink Real-Time Explorer

Interact with target computer and real-time application running on target computer

Description

Simulink Real-Time provides a single point of contact for configuring the development and target computers and interacting with a real-time application. You can monitor and trace signals, tune parameters, create and run instrument panels, and export and run an Explorer configuration as a standalone executable.

Note: Do not use Simulink external mode while Simulink Real-Time Explorer is running. Use only one interface or the other.

Use Simulink Real-Time Explorer for the following tasks. For examples, click the links in the **More Information** column.

Target Computer Configuration

Capability	More Information
Configure target computer communication parameters.	<ul style="list-style-type: none"> “PCI Bus Ethernet Setup” “USB-to-Ethernet Setup”
Configure target computer configuration settings.	“Target Computer Settings”
Configure target computer startup and create boot images for target computers.	“Target Computer Boot Methods”

Real-Time Application Access and Control

Capability	More Information
<ul style="list-style-type: none"> Connect target computers to a development computer, and then disconnect them. 	<ul style="list-style-type: none"> “Configure and Control a Real-Time Application” “Process a Real-Time Application with Simulink Real-Time Explorer”

Capability	More Information
<ul style="list-style-type: none"> • Load a prebuilt real-time application into a target computer, and then unload it. • Start and then stop running a real-time application that you downloaded to the target computer. • Display execution time, task execution time, and other properties of the real-time application. • Change stop time and sample times without regenerating code. 	

Signal Access

Capability	More Information
Filter and group hierarchical signals	“Display and Filter Hierarchical Signals and Parameters”
Monitor signals.	<ul style="list-style-type: none"> • “Monitor Signals with Simulink Real-Time Explorer” • “Instrument a Stateflow Subsystem”
<ul style="list-style-type: none"> • Add host, target, or file scopes for the downloaded real-time application, and then remove them. • Add signals to scopes, and then remove them. • Configure scope properties. 	<ul style="list-style-type: none"> • “Create Target Scopes with Simulink Real-Time Explorer” • “Create Host Scopes with Simulink Real-Time Explorer” • “Create File Scopes with Simulink Real-Time Explorer”

Capability	More Information
<ul style="list-style-type: none"> • Configure a host scope display and use it for viewing signal values. • Start and stop scopes. • Browse file scope output files on the target computer file system. 	
Create, save, and load signal groups.	“Create Signal Groups with Simulink Real-Time Explorer”

Parameter Tuning

Capability	More Information
Filter and group hierarchical parameters	“Display and Filter Hierarchical Signals and Parameters”
Display and tune parameter values for the signals while the real-time application is running.	<ul style="list-style-type: none"> • “Tune Parameters with Simulink Real-Time Explorer” • “Tune Parameter Structures with Simulink Real-Time Explorer”
Create, save, and load parameter groups.	“Create Parameter Groups with Simulink Real-Time Explorer”

Instruments and Instrument Panels

Capability	More Information
<p>Create, configure, save, and load graphical instrument panels for acquiring signals and tuning parameters.</p> <p>Attach parameters to instruments in instrument panels.</p> <p>Attach signals to instruments in instrument panels.</p>	<ul style="list-style-type: none"> • “Instrumentation for Real-Time Applications” on page 4-2 • “Instrument a Tank Model” on page 4-32

Capability	More Information
Start and stop instrument panels and use them to interact with real-time applications.	
Configure Scope instruments.	<ul style="list-style-type: none"> • “View Signal Waveforms with Scope Instrument” on page 4-19 • “View Signal Waveforms with Signal Triggered Scope Instrument” on page 4-27 • “Triggering Scope Instruments”

Explorer Configuration

Capability	More Information
Save and load environment properties, instrument panels, and Explorer configuration layouts.	<ul style="list-style-type: none"> • “Save Environment Properties” on page 4-40 • “Save and Load Instrument Panels” on page 4-41 • “Save and Restore Layouts” on page 4-42
Export Explorer configuration as a standalone executable.	“Export Explorer Configuration” on page 4-48
Run standalone executable generated from an exported Explorer configuration.	“Unpack and Run Standalone Configuration” on page 4-49

Windows

Simulink Real-Time Explorer provides tool windows and workspace windows in its main window. Tool windows include **Targets**, **Applications**, **Scopes**, **Palette**, **Panels**, **Output**, and so on. Workspace windows include windows for interfacing with the target application, setting environment properties, setting application and scope properties, and so on. You can resize tool windows and drag them by their title bar. You can drag workspace windows by their tab.

Layouts

To create optimal window layouts for your work, customize the position, size, and behavior of the various windows. When you customize the layout for the tool windows,

Explorer retains the positions, size, and docking locations of each tool window. For example, assume that you change the docking location of the **Targets** tool window, and then close Explorer and MATLAB. The next time that you start Explorer, the target tool window is docked in that same location. Workspace window customizations do not persist across different MATLAB sessions.

Tab Groups

To manage limited workspace while you are working with two or more Explorer windows, use Tab Groups. You can organize multiple workspace windows and tool windows into vertical and horizontal Tab Groups. You can drag windows from one Tab Group to another.

Arrange Windows

You can dock tool windows anywhere within the main window frame, float them as separate windows independent of the main window, and hide them. You can dock workspace windows only within the main frame. You cannot float them as separate windows or hide them.

You can arrange tool windows as follows:

- Pin tool windows to the left or the right of the tab well.
- Dock tool windows to the edge of main frame.
- Float tool windows over or outside the main window.
- Hide tool windows along the edge of the main window.
- Display tool windows on different monitors.
- Save window placement to a custom layout and restore it.

You arrange tool windows by dragging and dropping them or by right-clicking the window title bar and clicking a menu command.

Dock Windows with a Guide Diamond

When you click and drag a tool window title bar or a workspace window tab, a guide diamond appears. As you drag the window, place your cursor over one of the arrows in the diamond. A shaded area appears that shows where the window will appear after you release the mouse button.

To move a dockable window without snapping it into place, press the **Ctrl** key while you drag the window.

To return a tool window or workspace window to its most recent docked location, double-click the window title bar or window tab. You can fasten tool windows to one side of a frame in the main window. When you drag a tool window to another location, a guide diamond appears to help you redock the window.

Close and Auto Hide Tool Windows

You can close a tool window by clicking the **X** in the upper right of the title bar. To reopen the window, in the **View** menu, select the tool window that you want to reopen. If the tool window is closed, the tool window opens and becomes the active selected tool window. If the tool window is already open, it becomes the currently active tool window.

Tool windows support Auto Hide (⌘), which causes a window to slide out of the way when you use a different window. When you autohide a window, its name appears on a tab at the edge of the frame of the main window. To use the window again, place your cursor over the tab or select the tool window from the **View** menu. The window slides back into view.

Create and Save Layouts

You can create and save a custom layout that includes workspaces that you created for a specific target computer and real-time application. You can then switch between layouts with a single command. If you open a layout that contains workspaces, the target computer must have the same application and resources as when you originally created the layout.

Restore Layouts

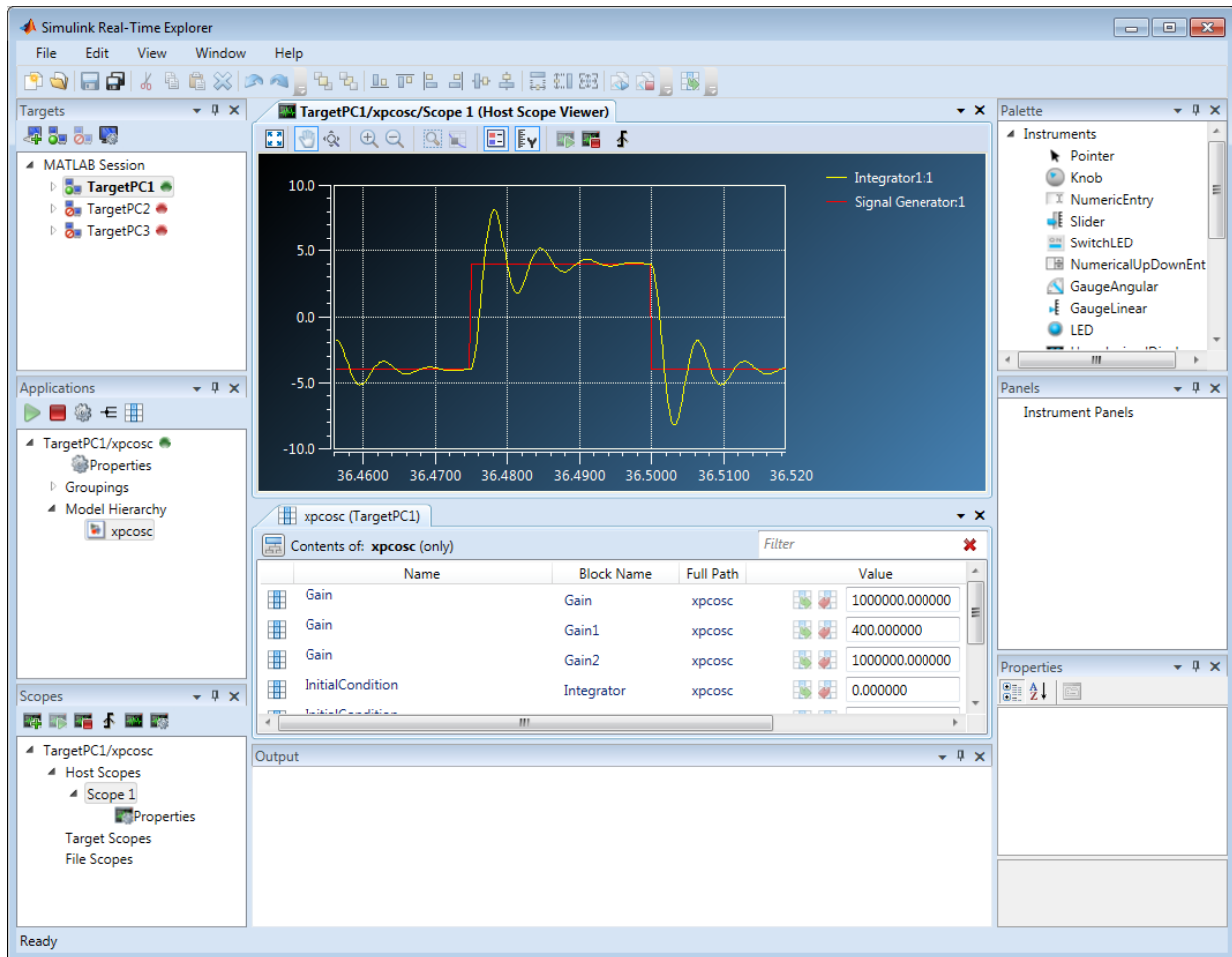
You can return Explorer to the original window layout for your settings by using the **Restore layout** command. When you run this command:

- Tool windows move to their original positions.
- Workspace windows that were previously opened close.
- Workspace windows included in the layout that are open in the original layout open.

You can restore workspace windows that depend on a specific target computer and real-time application. The target computer must be connected and contain the same application and resources as when you created the layout. Windows that fail to open produce an error message in the output window

Representative Layout

The figure shows a representative layout showing several tool and workspace windows.



In this figure, the tool windows are:

- **Targets** (top left) — Lists the targets in your Simulink Real-Time hierarchy. Under each target are nodes representing the properties and file system of the target.
- **Applications** (middle left) — Lists the real-time applications running on the targets. Under each application are nodes representing the properties, signal and parameter groupings, and model hierarchy of the application. If a target is not running a real-time application, it does not appear in the list.

- **Scopes** (bottom left) — Lists the scopes defined on the active real-time applications, whether predefined or dynamically created.
- **Output** (bottom center) — Displays status and error messages.
- **Properties** (bottom right) — Lists the properties of a selected instrument or layout element.
- **Panels** (middle right) — Lists the instrument panels that have been loaded.
- **Palette** (top right) — Lists the instrument and layout elements available for constructing instrument panels.

The workspace windows are:

- **TargetPC1/xpcosc/Scope 1 (Host Scope Viewer)** (top center) — The host scope viewer window shows signal and time information from a host scope that is running on the target computer.
- **xpcosc (TargetPC1)** (mid center) — The parameter window shows tunable parameters from the real-time application that is running on the target computer.

Exported Standalone Interface

You can export Simulink Real-Time Explorer as a standalone executable to a computer compatible with Windows[®]. You do not have to run MATLAB to run the standalone interface.

When you run Explorer as a standalone interface, it 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 computers. Before you export the Simulink Real-Time Explorer configuration, configure and test the communication parameters for each target computer.
- For each instrument, the exporting software records the real-time application and target computer environment with which it is associated.
- If you rename a target computer, to maintain the connection to the real-time application, update the **TargetName** parameter for each associated instrument.
- 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**.

Open the Simulink Real-Time Explorer

- In the Simulink Editor, click **Tools > Simulink Real-Time**.
- On a computer compatible with Windows, click the standalone executable.

Programmatic Use

In the Command Window, type `slrtexplr`.

See Also

Topics

“Signal Monitoring Basics”

“Signal Tracing Basics”

“Signal Logging Basics”

“File System Basics”

“Tunable Block Parameters and MATLAB Variables”

“Simulink Real-Time Scope Usage”

“Target Scope Usage”

“Host Scope Usage”

“File Scope Usage”

“Instrumentation for Real-Time Applications” on page 4-2

“Explorer Configuration Exported to Run Outside MATLAB” on page 4-16

Introduced in R2014a

Simulink Real-Time Explorer Instrumentation

- “Instrumentation for Real-Time Applications” on page 4-2
- “Explorer Configuration Exported to Run Outside MATLAB” on page 4-16
- “View Signal Waveforms with Scope Instrument” on page 4-19
- “View Signal Waveforms with Signal Triggered Scope Instrument” on page 4-27
- “Instrument a Tank Model” on page 4-32
- “Save Environment Properties” on page 4-40
- “Save and Load Instrument Panels” on page 4-41
- “Save and Restore Layouts” on page 4-42
- “Prepare Explorer Environment for Export” on page 4-43
- “Prepare Instrument Panel Configuration for Export” on page 4-45
- “Export Explorer Configuration” on page 4-48
- “Unpack and Run Standalone Configuration” on page 4-49

Instrumentation for Real-Time Applications

In this section...

“Instrument Selection and Binding” on page 4-4

“Display Instruments” on page 4-5

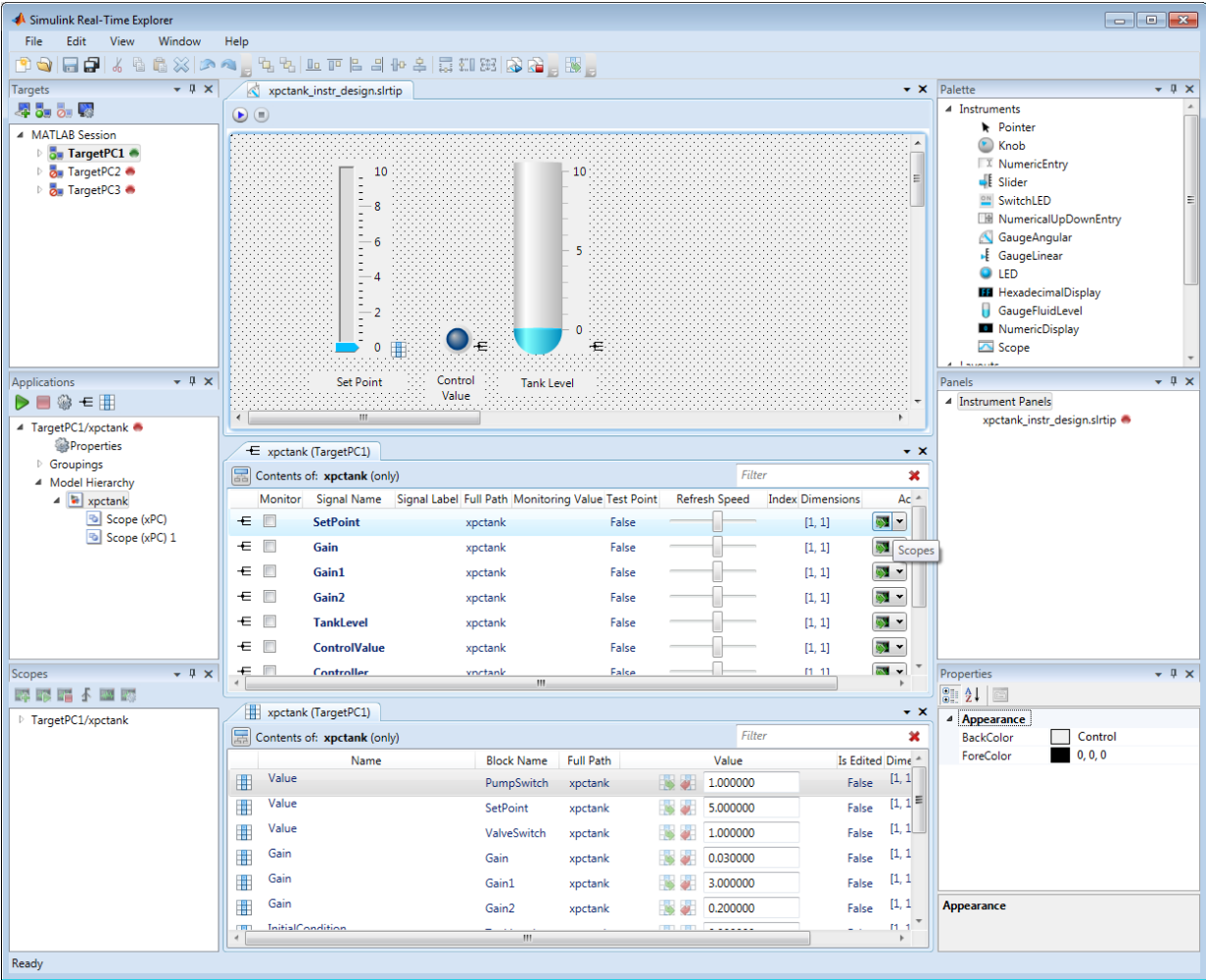
“Tuning Instruments” on page 4-7

“Layout Elements” on page 4-8

“Graphical Properties” on page 4-10








To visualize the behavior of a real-time application running on a target computer, Simulink Real-Time Explorer provides instrument panels. An instrument panel is an Explorer workspace into which you can insert one or more instruments. You can create and load instrument panels from the toolbar, from the **File** menu, and from the Panels tool 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 relationships among the instruments. 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.



At design time, you can manipulate the instruments by using the toolbar buttons.

Action	Icon	Notes
New, Open, Save, SaveAll		The available operations vary with the active window. <ul style="list-style-type: none"> You can save an instrument panel from

Action	Icon	Notes
		<p>within only the window for that panel.</p> <ul style="list-style-type: none"> You can create or open a signal or parameter group from within only the Applications window. You can save a group from within only the window for that group. You can create or open an instrument panel regardless of the active window. You can use the SaveAll button regardless of the active window.
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 real-time application, populate an instrument panel with instruments compatible with the parameters and signals that they represent.

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 and dropping it on 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 display instrument but not to a tuning instrument. You can bind a parameter to a tuning instrument and to a display instrument.

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

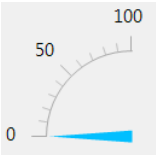
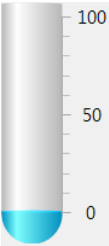
When you bind a parameter to a tuning or display instrument, the **Parameter** button  appears next to the instrument.




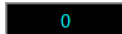
When the instrument panel is running, you can tune parameters by using tuning instruments. The software transmits the parameter changes to the real-time application. You can view the changed behavior by using display instruments.

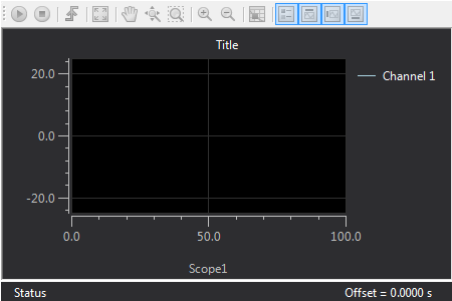
When the instrument panel is not running, you can add, remove, and lay out instruments and connect signals and parameters to them.

Display Instruments

The table shows some of the tasks that you can do with display instruments.

Action	Requiring	Use
<ul style="list-style-type: none"> Show the pressure in a container. Show the speed of a vehicle. Show current or voltage in a circuit. 	<ul style="list-style-type: none"> Real-valued data Show approximate value by angular displacement 	 <p>GaugeAngular</p>
<ul style="list-style-type: none"> Show the level of fluid in a container. Show the pressure in a pipe. 	<ul style="list-style-type: none"> Real-valued data Show approximate value by vertical displacement 	

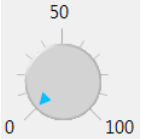
Action	Requiring	Use
<ul style="list-style-type: none"> Show the pressure in a container. Show audio output power. Show current or voltage in a circuit. 	<ul style="list-style-type: none"> Real-valued data Show approximate value by linear displacement 	<p>GaugeFluidLevel</p> 
<ul style="list-style-type: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

Action	Requiring	Use
<ul style="list-style-type: none"> Show a time-varying temperature measurement. Show a time-varying voltage measurement. 	<ul style="list-style-type: none"> Real-valued data Show time-varying waveforms 	

Scope

Tuning Instruments

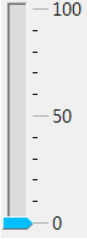

The table shows some of the tasks you can do with tuning instruments.

Action	Requiring	Use
<ul style="list-style-type: none"> Control gain of a radio receiver. Control amplification of a radio transmitter. 	<ul style="list-style-type: none"> Real-valued data Set approximate value by angular displacement 	
<ul style="list-style-type: none"> Enter initial set point control value for thermostat. Enter seed value for random number generator. 	<ul style="list-style-type: none"> Real-valued data Set exact numeric value 	<input type="text" value="0"/>
<ul style="list-style-type: none"> Control car radio audio volume using step-increment. Smoke test controller range in small number of clicks. 	<ul style="list-style-type: none"> Real-valued data Set initial value and step increment 	<input type="text" value="0"/>

Knob

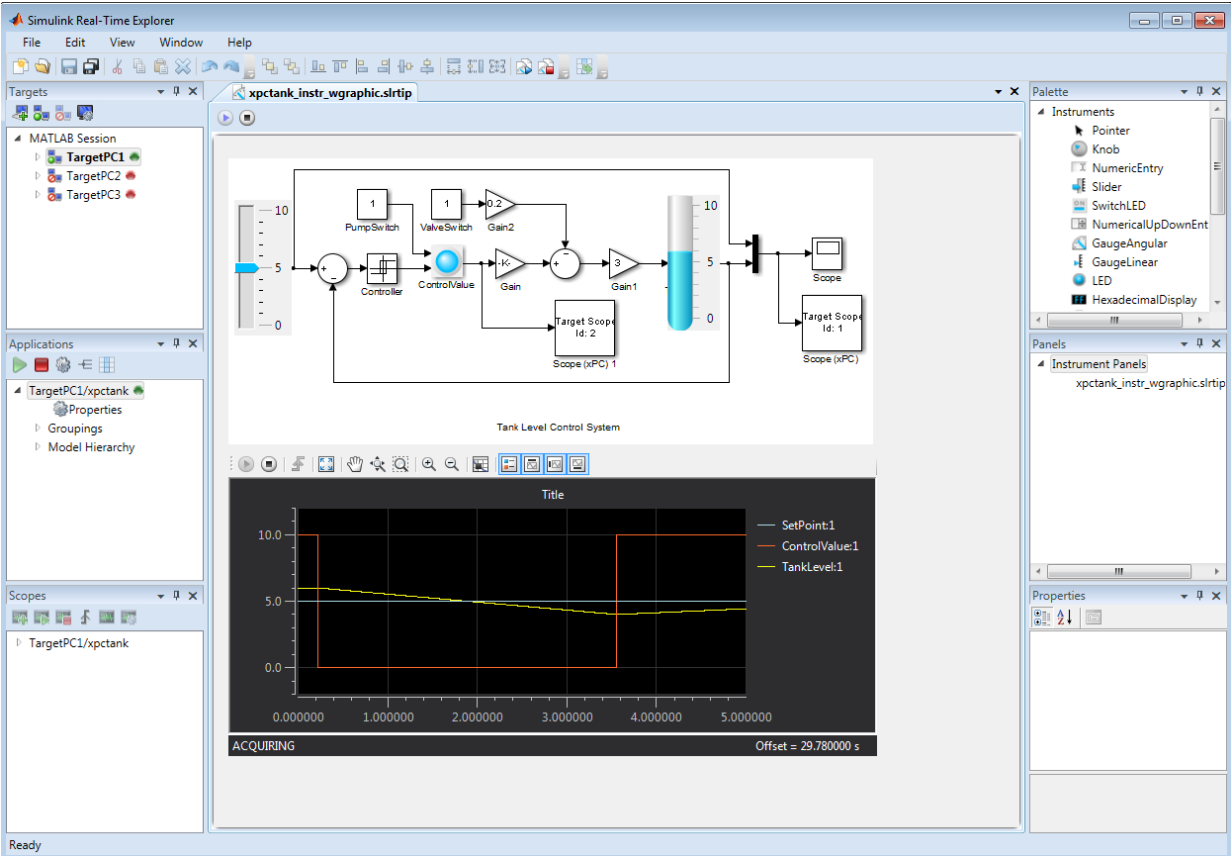
NumericEntry

NumericUpDownEntry

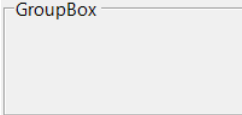
Action	Requiring	Use
<ul style="list-style-type: none"> Control frequency of a radio receiver. Control pressure valve setting. 	<ul style="list-style-type: none"> Real-valued data Set approximate values by linear displacement 	 <p>Slider</p>
<ul style="list-style-type: none"> Turn on a power supply. Close a gate valve. 	<ul style="list-style-type: none"> Boolean data Turn control on or off 	 <p>SwitchLED</p>


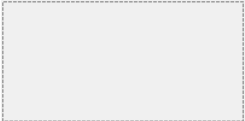
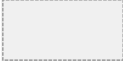
Layout Elements

To make the relationships among the instruments clearer, you can add layout elements, such as a picture box. In the figure, a picture box shows to which signals the instruments are bound.



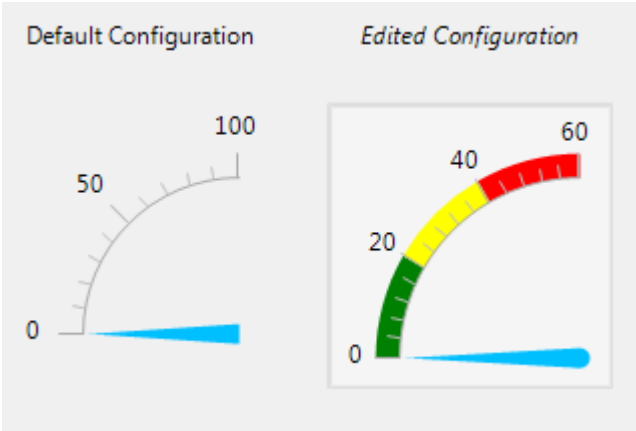
The table shows some ways that you can modify your layout with layout elements.


Action	Requiring	Use
<ul style="list-style-type: none">Group electrical instruments together and label the group.Group pressure instruments together and label the group.	<ul style="list-style-type: none">Design-time box resizingRun-time static display	 GroupBox


Action	Requiring	Use
<ul style="list-style-type: none"> Label an electrical instrument. Label a pressure instrument. 	<ul style="list-style-type: none"> Design-time box resizing, left-right and up-down text alignment 	 <p>Label</p>
<ul style="list-style-type: none"> Group the electrical group with the pressure group and scroll between the groups. 	<ul style="list-style-type: none"> Run-time static display Design-time box resizing Run-time box scrolling 	 <p>Panel</p>
<ul style="list-style-type: none"> Group a picture box with a group of instruments to show what the instruments are measuring. 		
<ul style="list-style-type: none"> Insert an image of a circuit diagram on a panel behind electrical instruments. Insert an image of a circulation diagram on a panel behind pressure instruments. 	<ul style="list-style-type: none"> Design-time image stretch, zoom, center, and autosize Run-time static display 	 <p>PictureBox</p>

Graphical Properties

You can configure the graphical properties of the instruments. The figure shows an instrument in its default configuration and with selected property changes.



To access a dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

Instrument Properties

The table shows some of the ways that you can modify the appearance of your instruments through the **Instrument** properties.

Action	Use
Set instrument value range	ScaleRange <ul style="list-style-type: none">• Min, Span — Value bounds. Max is read-only.• Reverse — If True, reverse direction of display• ScaleType — One of Linear, Log10, SplitLinearLog10.• SplitStart, SplitPercent — Value and percentage of scale split for ScaleType SplitLinearLog10 only.• AngleMin, AngleSpan — Angular value bounds. AngleMax is read-only. For GaugeAngular and Knob only.
Set value display properties	ScaleDisplay > GeneratorStyle > Auto

Action	Use
	<p>In ScaleDisplay > GeneratorAuto:</p> <ul style="list-style-type: none"> • MidIncluded — If True, insert a tick halfway between major ticks. <p>If MinorCount is even, space the minor ticks equally around the center tick. If MinorCount is odd, replace the center tick with the middle tick.</p> <ul style="list-style-type: none"> • FixedMinMaxMajor — If True, the top and bottom ticks are constrained to be major ticks with min/max values defined by Min and Span. • MinorCount — Number of minor ticks between major ticks. • MinTextSpacing — Minimum space between scale ticks. • DesiredIncrement — 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. <p>You can also set properties for ScaleDisplay > GeneratorStyle > Fixed and ScaleDisplay > GeneratorStyle > Custom.</p> <p>ScaleDisplay > TextFormatting</p> <ul style="list-style-type: none"> • Style — One of Number, Thousands, Prefix, Exponent, Price32nds, DateTime, DateTimeUTC. • PrecisionStyle — One of FixedDecimalPoints, SignificantDigits, None. • Precision — Number of digits to the right of the decimal point. • UnitsText — Text to display next to tick labels.

Action	Use
Set pointer properties	<p>Pointer</p> <ul style="list-style-type: none"> • Style — The pointer style depends upon the instrument. <ul style="list-style-type: none"> • GaugeAngular — One of Arrow, ArrowLine, Line, Triangle. • GaugeLinear — One of Pointer, Triangle, TLine, ColorBar, Tube. • Knob — One of Dot, DotRaised, DotSunken, LineCenter, LineCustom, Triangle. • Slider — One of Bar, BarIndicator, BarIndicatorLight, Pointer • Size — The width of the pointer, in pixels. • Margin — The margin around the pointer, in pixels.
Set border properties	<p>Border</p> <ul style="list-style-type: none"> • Style — One of Bump, Etched, Flat, Raised, RaisedInner, RaisedOuter, Sunken, SunkenInner, SunkenOuter, RoundedSides. • Margin — Border margin, in pixels • ThicknessDesired — Specified thickness, in pixels
Set font properties	<p>ScaleDisplay > TickMajor > Font</p> <p>ScaleDisplay > TickMid > Font</p> <ul style="list-style-type: none"> • Name — Font for text display. • Font style — One of Light, Semilight, Regular, Italic, Semibold, Bold, Bold Italic, Light Oblique, Semibold Oblique. • Size — Font size, in points • Strikeout — If True, add strikeout effect. • Underline — If True, add underline effect.

Action	Use
Set color properties	<p>ColorSections</p> <p>By default, the list of color sections is empty. To add color sections, open the Color Section Collection Editor dialog box and click Add.</p> <ul style="list-style-type: none"> • Color — Open the Color dialog box, which contains the color palettes Custom, Web, and System. • Start — Start of color section, in range units. • Stop — End of color section, in range units. <p>Color properties are associated with the following instrument properties: Border, Pointer, ScaleDisplay > TickMajor, ScaleDisplay > TickMid, ScaleDisplay > TickMinor.</p>

Appearance Properties

The table shows some of the ways that you can modify the appearance of your instruments through the **Appearance** properties.

Action	Use
Set font properties	<ul style="list-style-type: none"> • Name — Font for text display. • Font style — One of Light, Semilight, Regular, Italic, Semibold, Bold, Bold Italic, Light Oblique, Semibold Oblique. • Size — Font size, in points • Strikeout — If True, add strikeout effect. • Underline — If True, add underline effect.
Set color properties	<p>BackColor, ForeColor — Open the color palettes Custom, Web, and System.</p>
Set image properties	<ul style="list-style-type: none"> • BackgroundImage — Navigate to a background image file. • BackgroundImageLayout — One of Tile, None, Center, Stretch, Zoom.

See Also

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

Related Examples

- “View Signal Waveforms with Scope Instrument” on page 4-19
- “View Signal Waveforms with Signal Triggered Scope Instrument” on page 4-27
- “Instrument a Tank Model” on page 4-32
- “Triggering Scope Instruments”

More About

- Simulink Real-Time Explorer

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.

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:

- Check that the candidate computer is compatible with 64-bit Windows.
- Check that the CPU and operating system meet the requirements for executing the standalone Simulink Real-Time Explorer executable.
- Check 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:

- Check 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**.

Limitations as Standalone Executable

When you export Simulink Real-Time Explorer as a standalone executable, it 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 computers. Before you export the Simulink Real-Time Explorer configuration, configure and test the communication parameters for each target computer.
- For each instrument, the exporting software records the real-time application and target computer environment with which it is associated.
- If you rename a target computer, to maintain the connection to the real-time application, update the **TargetName** parameter for each associated instrument.

- 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**.

More About

- “Standalone Boot Method”

View Signal Waveforms with Scope Instrument

In this section...

“Create Instrument Panel” on page 4-20

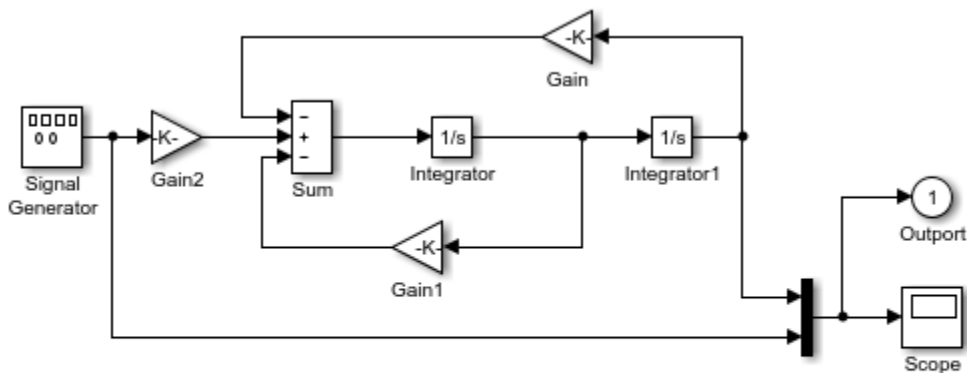
“Configure Slider Instrument for Parameter Tuning” on page 4-20

“Configure Scope Instrument with Default Triggering” on page 4-22

“Run Instrumented Model” on page 4-23

In this example, create an instrument panel for the `xpcosc` model that contains the following instruments:

- Slider — To tune the input signal amplitude (Signal Generator/Amplitude).
- Slider — To tune the oscillator feedback damping parameter (Gain1/Gain).
- Scope — To display the input signal (Signal Generator) and the oscillator output (Integrator1).




Model `xpcosc`
Simulink Real-Time example model

Copyright 1999-2013 The MathWorks, Inc.

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.

Create Instrument Panel

Create and save an instrument panel for the `xpcosc` model.

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

Configure Slider Instrument for Parameter Tuning

Select and configure an instrument to tune two parameters in the `xpcosc` model. You must have previously created the `xpcosc_scope_instr_freerun.slrtip` instrument panel.

The parameter characteristics are listed in this table.

Name	Type	Range	Purpose
Amplitude	Numeric	0–10 units	Represents the amplitude of the input signal Signal Generator. You do not have to set it to an exact value.
Gain1	Numeric	0–1000 units	Represents the damping of the oscillator feedback signal Gain1. You do not have to set it to an exact value.



The Slider instrument meets the requirement for Amplitude and Gain1. To set exact numeric values, 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 `xpcosc_scope_instr_freerun.slrtip`.
- 2 Select the instrument.

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

- 3 Bind the parameter to the instrument.

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

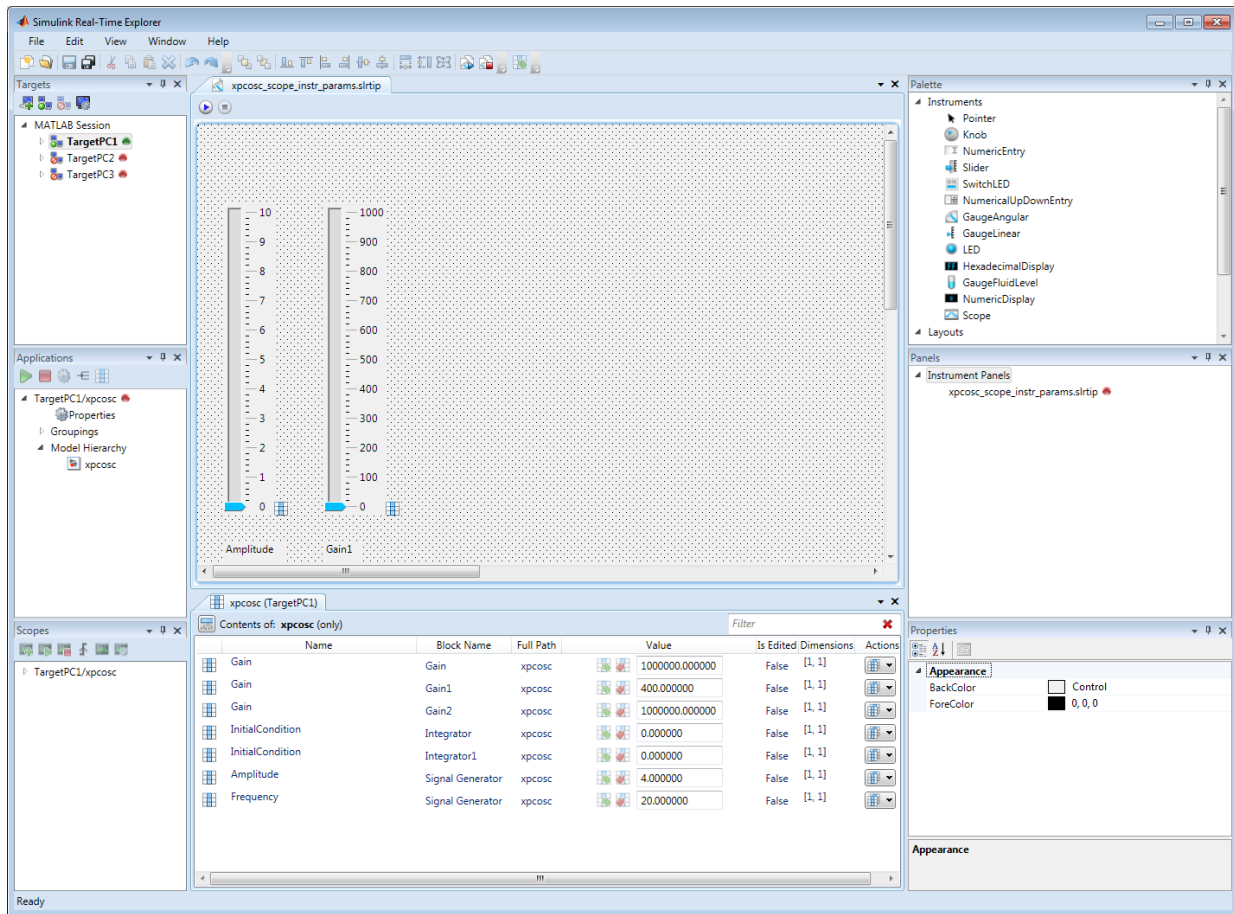
- 10 Repeat steps 2–9 for parameter `Gain1`.

Set the property **Min** to 0 and property **Span** to 1000.

Set the **Text** property to `Gain1`.

- 11 Click the **Save** button .

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



Configure Scope Instrument with Default Triggering

Add a Scope instrument and configure it with the default triggering, which is a TriggerMode of SINGLESHOT and a TriggerSource of FREERUN. You must have previously created the xpcosc_scope_instr_freerun.slrtip instrument panel.

The signal characteristics are listed in this table.

Name	Type	Range	Purpose
Signal Generator	Numeric	-25:25 units	Represents the time-varying value of the signal generator input.
Integrator1	Numeric	-25:25 units	Represents the time-varying value of the oscillator output.

To select and configure the Scope instrument:

- 1 Load the instrument panel.

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

- 2 Select the instrument.

From the **Palette** pane, drag a Scope instrument to the `xpcosc_scope_instr_freerun.slrtip` instrument panel.

- 3 Bind the signals to the instrument.

To bind the Signal Generator signal to the Scope instrument, open the Signal workspace for model `xpcosc` (☰ on the toolbar). Click the **Signal** icon ☰ next to signal Signal Generator and drag the icon to the Scope instrument.

The name `Signal Generator:1` appears in the Scope instrument legend.

- 4 In a similar manner to step 3, bind the Integrator1 signal to the Scope instrument
- 5 Set the instrument range.

Select the Scope instrument, and then click the **YAxesLimits** node in the **Properties** list.

- 6 Set property **Max** to 25 and **Min** to -25.




- 7 Click the **Save** button .

Run Instrumented Model

Run the instrumented `xpcosc` model, tune the parameters, and view the output waveform. You must have previously built and downloaded the `xpcosc` model and configured the `xpcosc_scope_instr_freerun.slrtip` instrument panel.

- 1 Load the instrument panel.


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

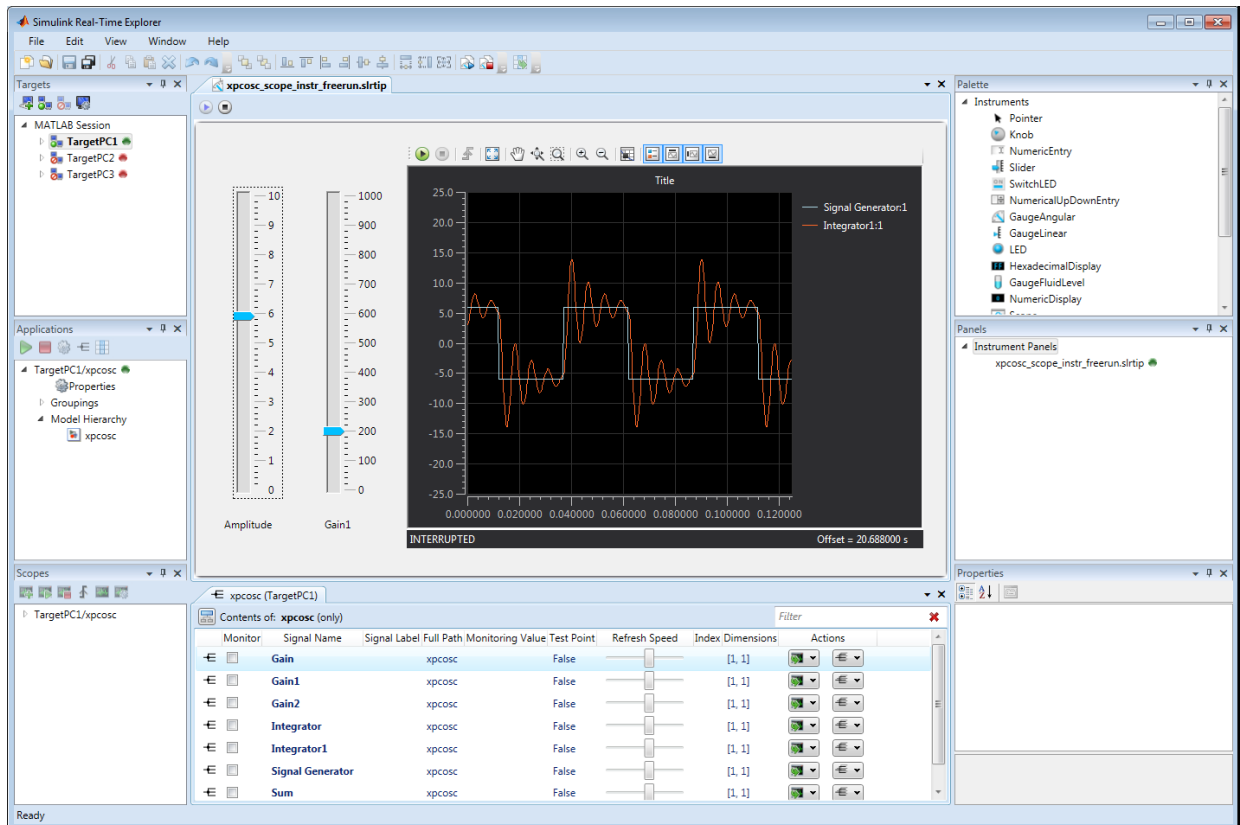
- 2 Set property **Stop time** to `inf` in the **Applications** pane ( on the toolbar).
- 3 To start the instrument, in the `xpcosc_scope_instr_freerun.slrtip` instrument panel, click the **Run Instrument** button .
- 4 To start execution, in the **Applications** pane, click the real-time application, and then click the **Start** button  on the toolbar.



The Scope instrument displays one set of captured waveforms, and then stops.

- 5 To change the amplitude and damping value, use the **Amplitude** and **Gain1** instruments.

For example, to increase the amplitude and reduce the oscillator damping, set **Amplitude** to `6` and **Gain1** to `200`.

- 6 To view the result of the parameter changes, click the **Start Acquisition** button  on the Scope toolbar.



- 7 To stop execution, in the **Applications** pane, click the real-time application, and then click the **Stop** button  on the toolbar.
- 8 To stop the instruments, in the `xpcosc_scope_instr_freerun.slrtip` instrument panel, click the **Stop Instrument** button .

See Also

Scope | Slider

More About

- “View Signal Waveforms with Signal Triggered Scope Instrument” on page 4-27

- “Instrument a Tank Model” on page 4-32
- “Triggering Scope Instruments”
- “Save and Load Instrument Panels” on page 4-41
- “Save and Restore Layouts” on page 4-42
- “Instrumentation for Real-Time Applications” on page 4-2
- “Display and Filter Hierarchical Signals and Parameters”

View Signal Waveforms with Signal Triggered Scope Instrument

The default trigger mode of the Scope instrument is **SINGLESHOT**, which displays one sample sweep. To trace a waveform in a continuously updated moving display, you can change the trigger mode to **REPEATED**. However, a moving display is often difficult to evaluate.

In this example, to produce a continuously updated static display, we configure the scope instrument to trigger mode **REPEATED** with trigger source **Signal**.

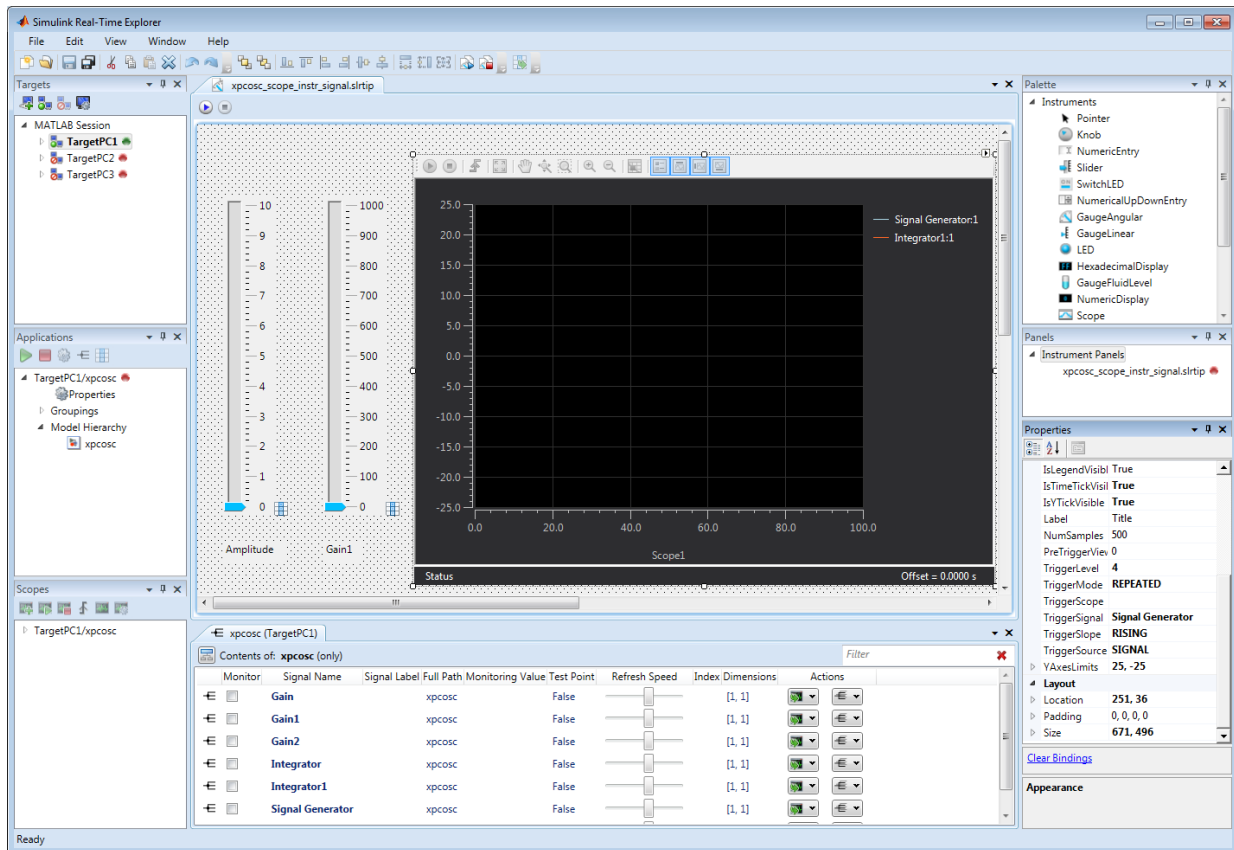
Preconditions

You must have previously built and downloaded the `xpcosc` model and configured the `xpcosc_scope_instr_freerun.slrtip` instrument panel. See “View Signal Waveforms with Scope Instrument” on page 4-19.

Configure Scope Instrument

- 1 Load the instrument panel.

In the **Panels** pane, right-click the **Instrument Panels** node and select **Existing**. From the list, select `xpcosc_scope_instr_freerun.slrtip`.
- 2 Select the Scope instrument.
- 3 Open the **Instrument** properties list.
- 4 Set **TriggerMode** to **REPEATED**.
- 5 Set **TriggerSource** to **Signal**.
- 6 Set **TriggerSignal** to **Signal Generator**.
- 7 Set **TriggerSlope** to **Rising**.
- 8 Set **TriggerLevel** to **4**.
- 9 Save the instrument panel under another name, such as `xpcosc_scope_instr_signal.slrtip`.





Run Instrumented Model

Run the instrumented `xpcosc` model, tune the parameters, and view the output waveform with signal triggering and forced triggering. You must have previously built and downloaded the `xpcosc` model and configured the `xpcosc_scope_instr_signal.slrtrip` instrument panel.

- 1 Load the instrument panel.

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

- 2 Set property **Stop time** to `inf` in the **Applications** pane (⚙️ on the toolbar).

- 3 To start the instruments, in the `xpcosc_scope_instr_signal.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.

The Scope instrument displays a continuous series of captured waveforms.


- 5 To change the amplitude and damping value, use the **Amplitude** and **Gain1** instruments.

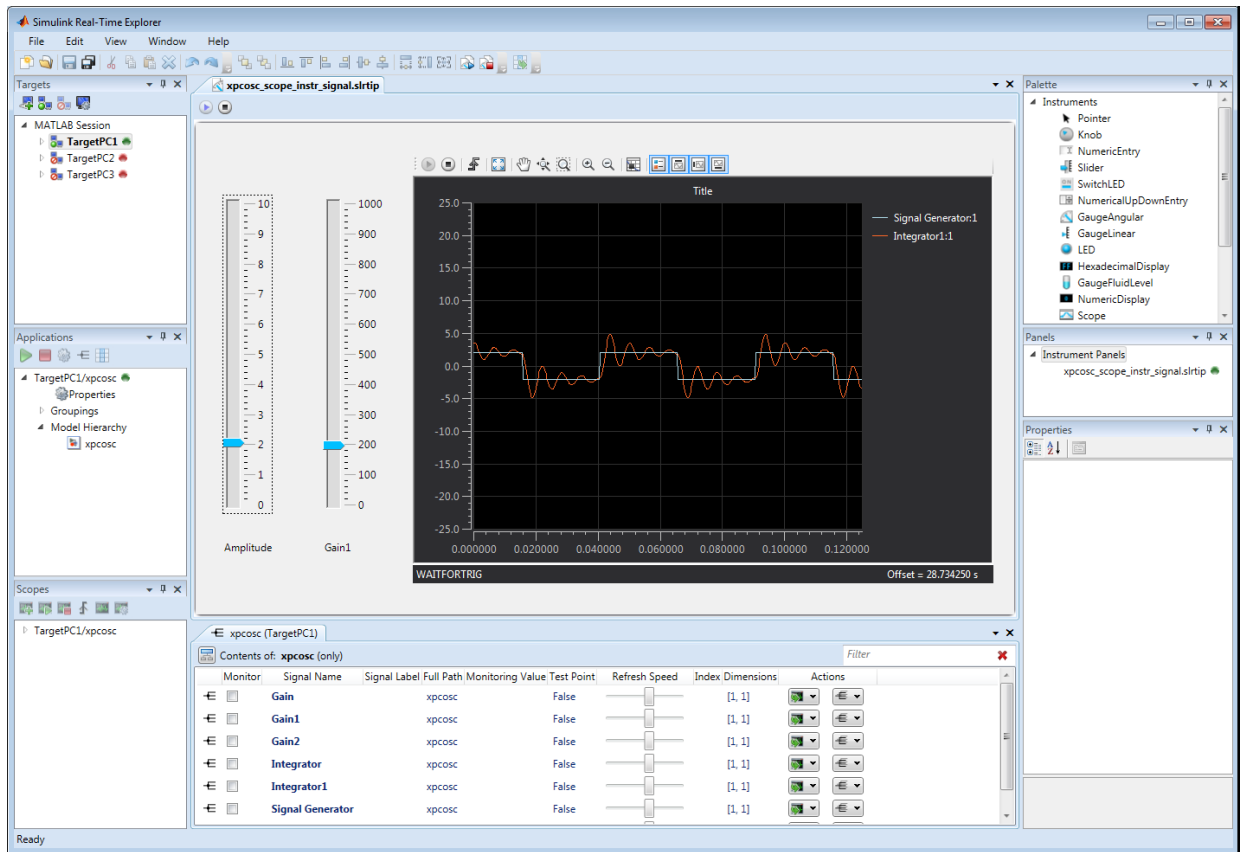
For example, to increase the amplitude and reduce the oscillator damping, set **Amplitude** to 6 and **Gain1** to 200.



The Scope instrument displays the changes in waveforms immediately.

- 6 Reduce the amplitude below the **TriggerLevel** (4).

The Scope instrument stops displaying the changes in waveforms because the signal does not have a high enough amplitude to reach the trigger level.

- 7 To display waveforms with an amplitude below the trigger level, click the **Force Trigger** button  on the toolbar.



- 8 To stop execution, in the **Applications** pane, click the real-time application, and then click the **Stop** button  on the toolbar.
- 9 To stop the instruments, in the `xpcosc_scope_instr_signal.slrtip` instrument panel, click the **Stop Instrument** button .

See Also

Scope | Slider

More About

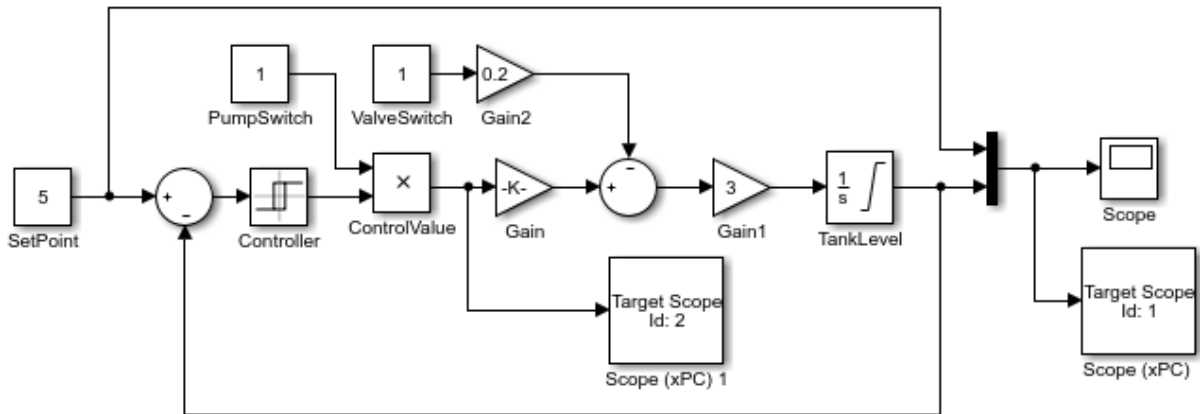
- “View Signal Waveforms with Scope Instrument” on page 4-19

- “Instrument a Tank Model” on page 4-32
- “Triggering Scope Instruments”
- “Save and Load Instrument Panels” on page 4-41
- “Save and Restore Layouts” on page 4-42
- “Instrumentation for Real-Time Applications” on page 4-2
- “Display and Filter Hierarchical Signals and Parameters”

Instrument a Tank Model

This example shows how to create an instrument panel for the `xpctank` model that contains the following instruments:

- Slider — To tune the required tank level (SetPoint).
- GaugeFluidLevel — To display the actual tank level (TankLevel).
- LED — To display the pump control status (ControlValue).



Tank Level Control System

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.

Create Instrument Panel

In this step, you create and save an instrument panel for the `xpctank` model.

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.

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.

Parameter Characteristics

The parameter characteristics for `SetPoint` are:

- Type — Numeric
- Range — 0–10 units
- Purpose — 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 requirements for `SetPoint`. To set an exact numeric value, use, for example, a `NumericEntry` instrument.

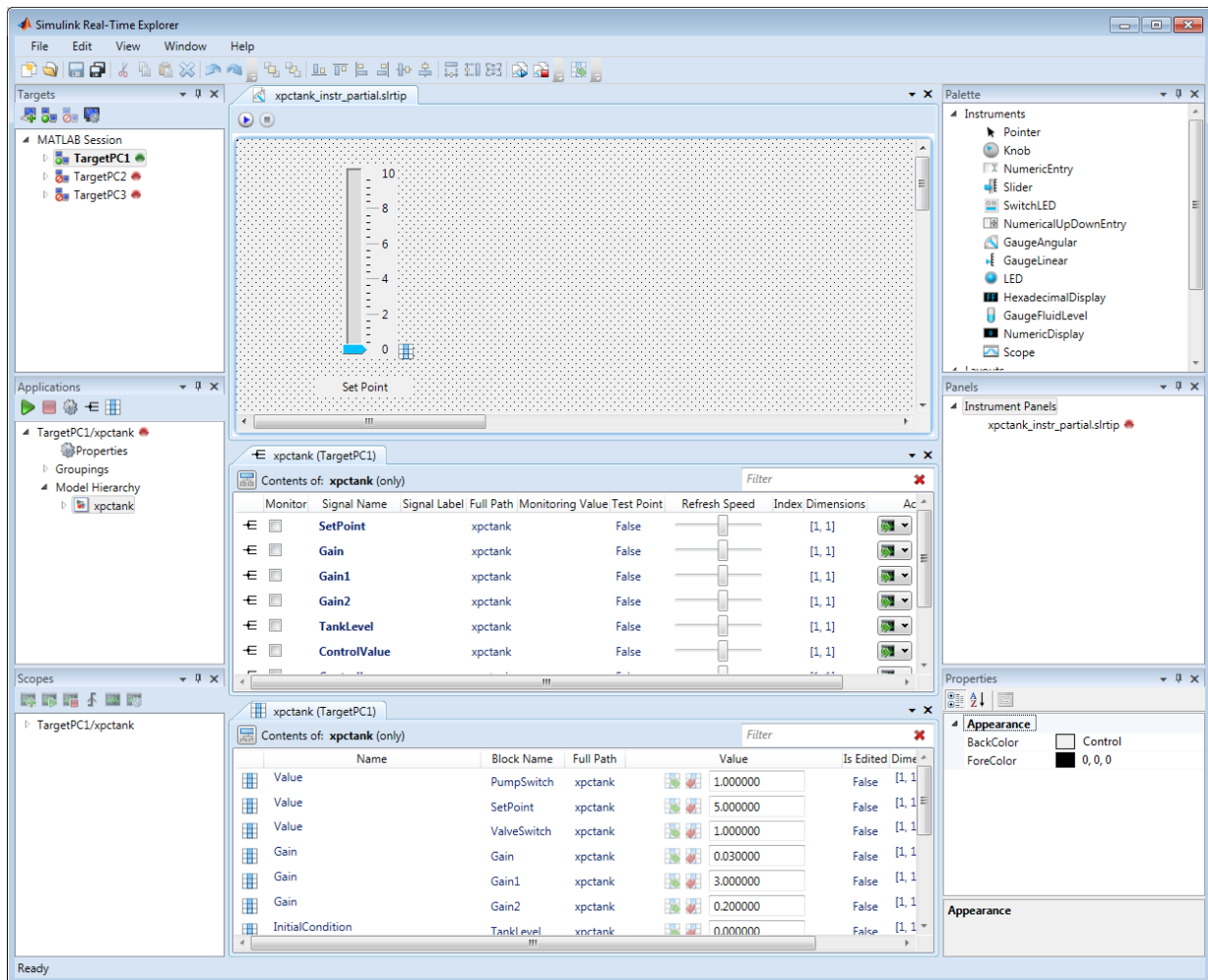
Configure Slider Instrument

To select and configure the Slider 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`. 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.



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.

Signal Characteristics The signal characteristics for TankLevel are:

- Type — Numeric
- Range — 0–10 units
- Purpose — Represents the current tank fluid level. You do not have to display an exact value.

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

The signal characteristics for ControlValue are:

- Type — Boolean
- Range — 1, 0
- Purpose — Represents the state of the pump (on or off).

The LED instrument meets the requirements for ControlValue.

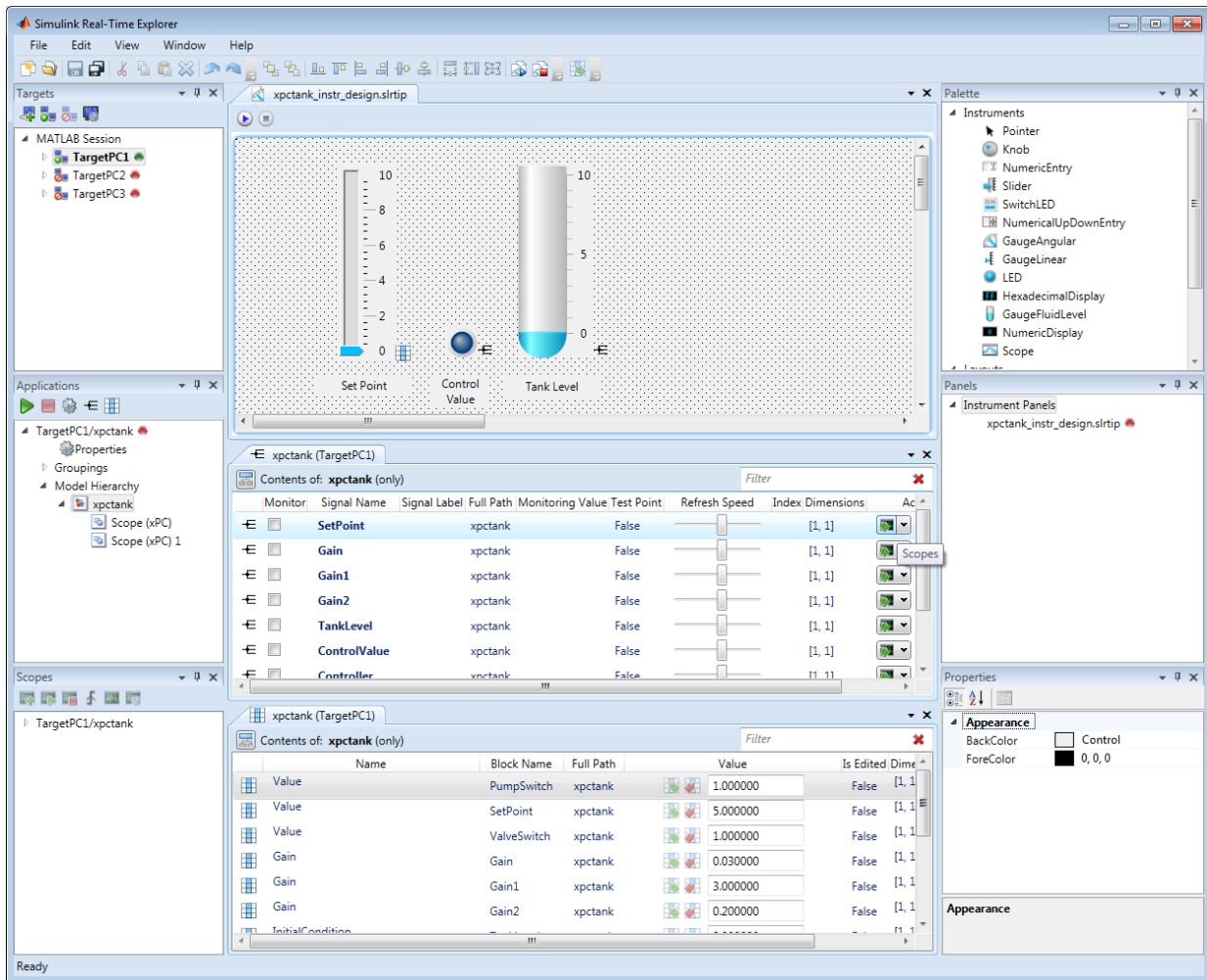
Configure Instruments

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. 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.

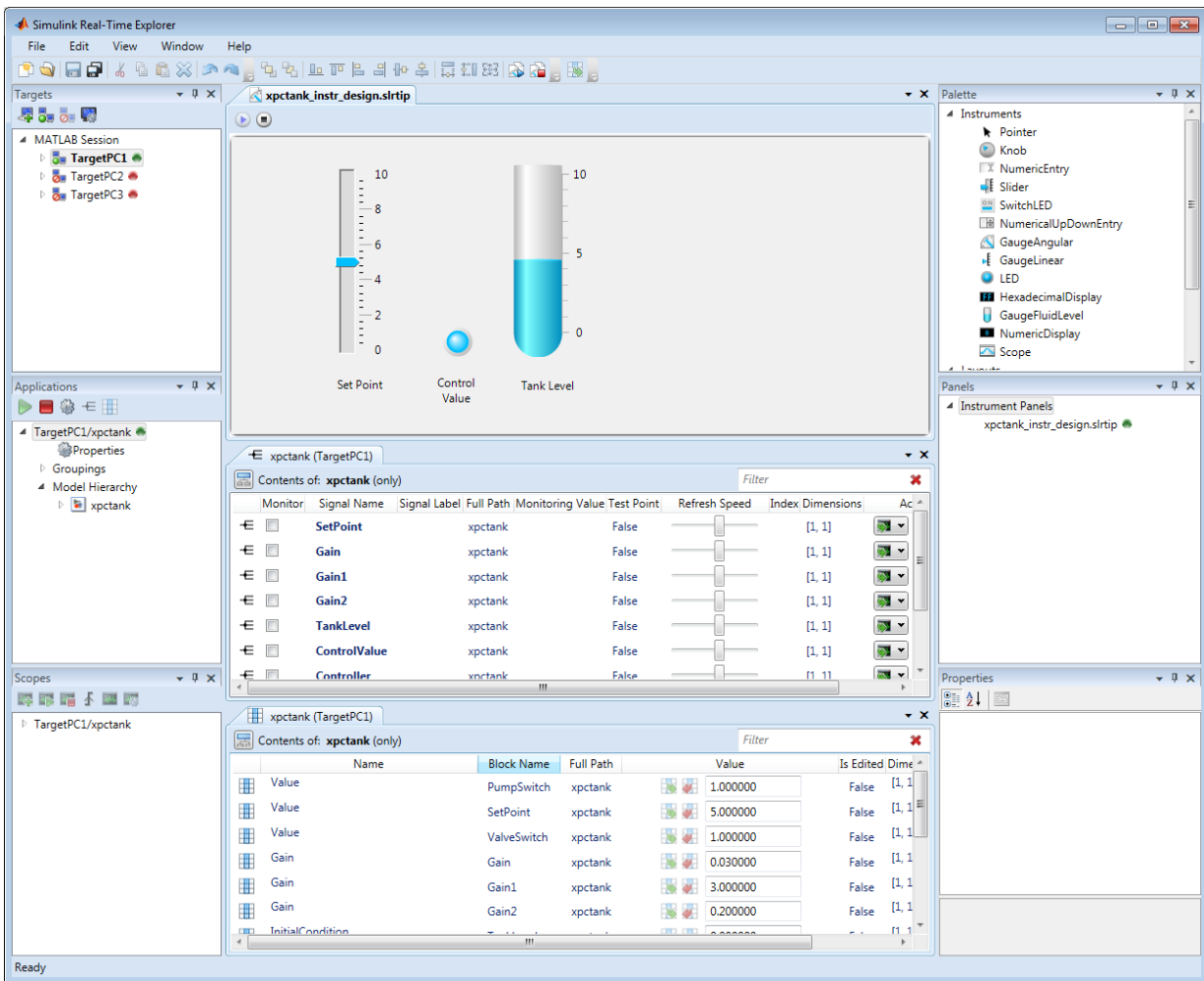


Run Instrumented Model

This example shows how to run the instrumented xpctank model. You must have previously built and downloaded the xpctank model and configured the xpctank_instr_design.slrtip instrument panel.

- 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.
- 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.
- 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.



- 1 To stop execution, in the **Applications** pane, click the real-time application, and then click the **Stop** button.
- 2 To stop the instruments, in the `xpctank_instr_design.slrtip` instrument panel, click the **Stop Instrument** button.

See Also

GaugeFluidLevel | LED | Slider

More About


- “View Signal Waveforms with Scope Instrument” on page 4-19
- “View Signal Waveforms with Signal Triggered Scope Instrument” on page 4-27
- “Save and Load Instrument Panels” on page 4-41
- “Save and Restore Layouts” on page 4-42
- “Instrumentation for Real-Time Applications” on page 4-2
- “Display and Filter Hierarchical Signals and Parameters”

Save 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**.

Prepare Explorer Environment for Export

Check 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 with Simulink Real-Time Explorer”.

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

- 1 Check that the candidate computer is compatible with 64-bit Windows.
- 2 Check that the CPU and operating system meet the requirements for executing the standalone Simulink Real-Time Explorer executable.
- 3 Check 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 Check that the target computer CPU and operating system meet the requirements for running the Simulink Real-Time kernel.
- 2 Check 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 Check 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 Check 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. Check that the target computer loads the Simulink Real-Time kernel and starts the real-time application.

More About

- “Development Computer Requirements”
- “Target Computer Requirements”
- “PCI Bus Ethernet Setup”
- “USB-to-Ethernet Setup”
- “Target Computer Settings”
- “Standalone Boot Method”
- “Explorer Configuration Exported to Run Outside MATLAB” on page 4-16

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.

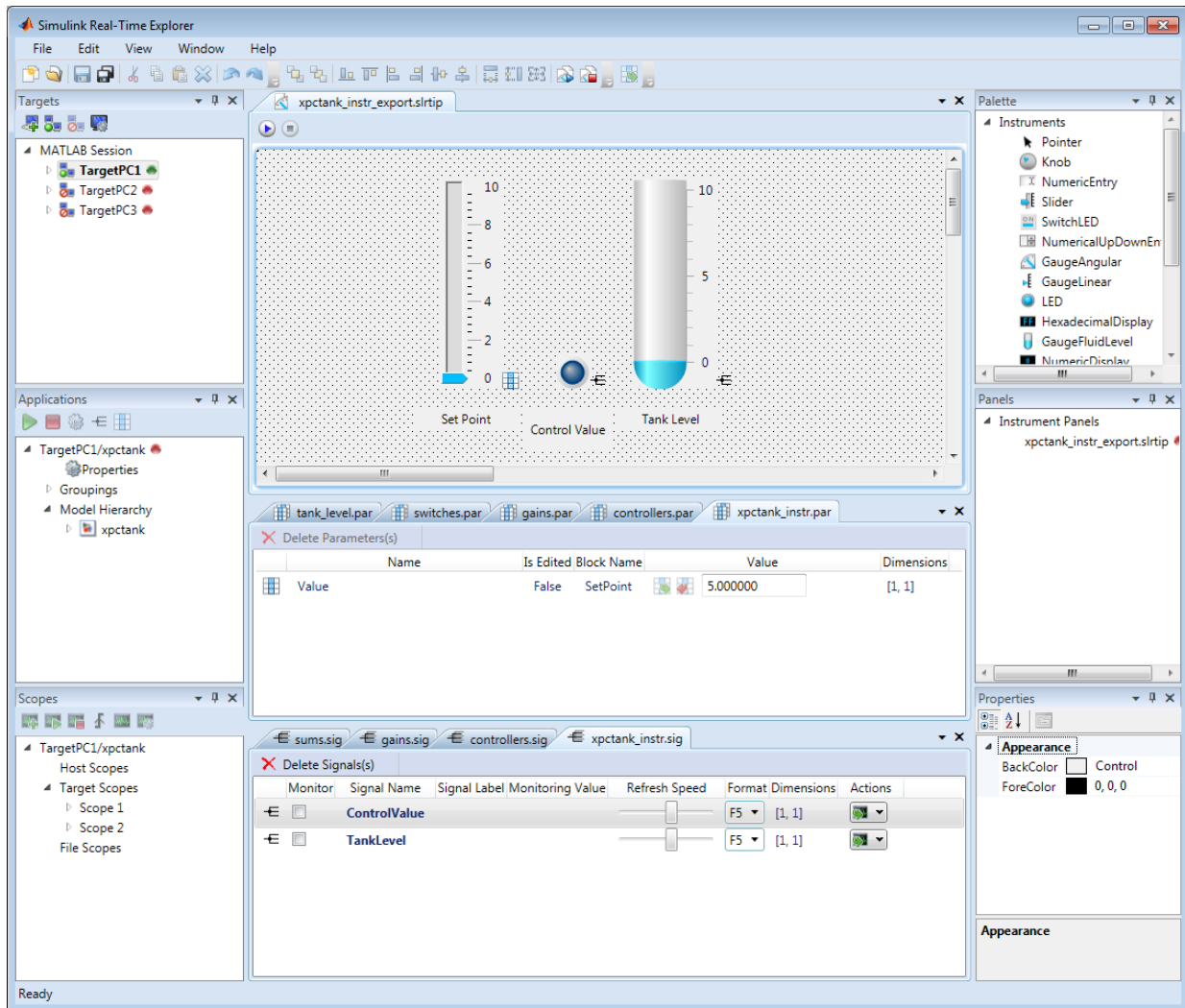
Note: When you run the standalone executable, you cannot access the model hierarchy. You can access only instrument panels and windows that were open when you exported the configuration. You can access only signals and parameters that were loaded in signal and parameter groups when you exported the configuration.

- 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 4-48.

More About

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

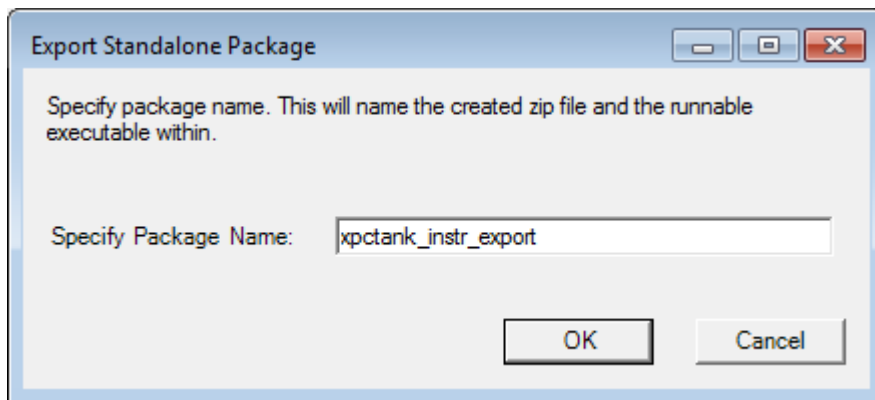
- “Instrumentation for Real-Time Applications” on page 4-2

Export Explorer Configuration

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 4-45.

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 4-49.

More About

- “Instrumentation for Real-Time Applications” on page 4-2

Unpack and Run Standalone Configuration

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

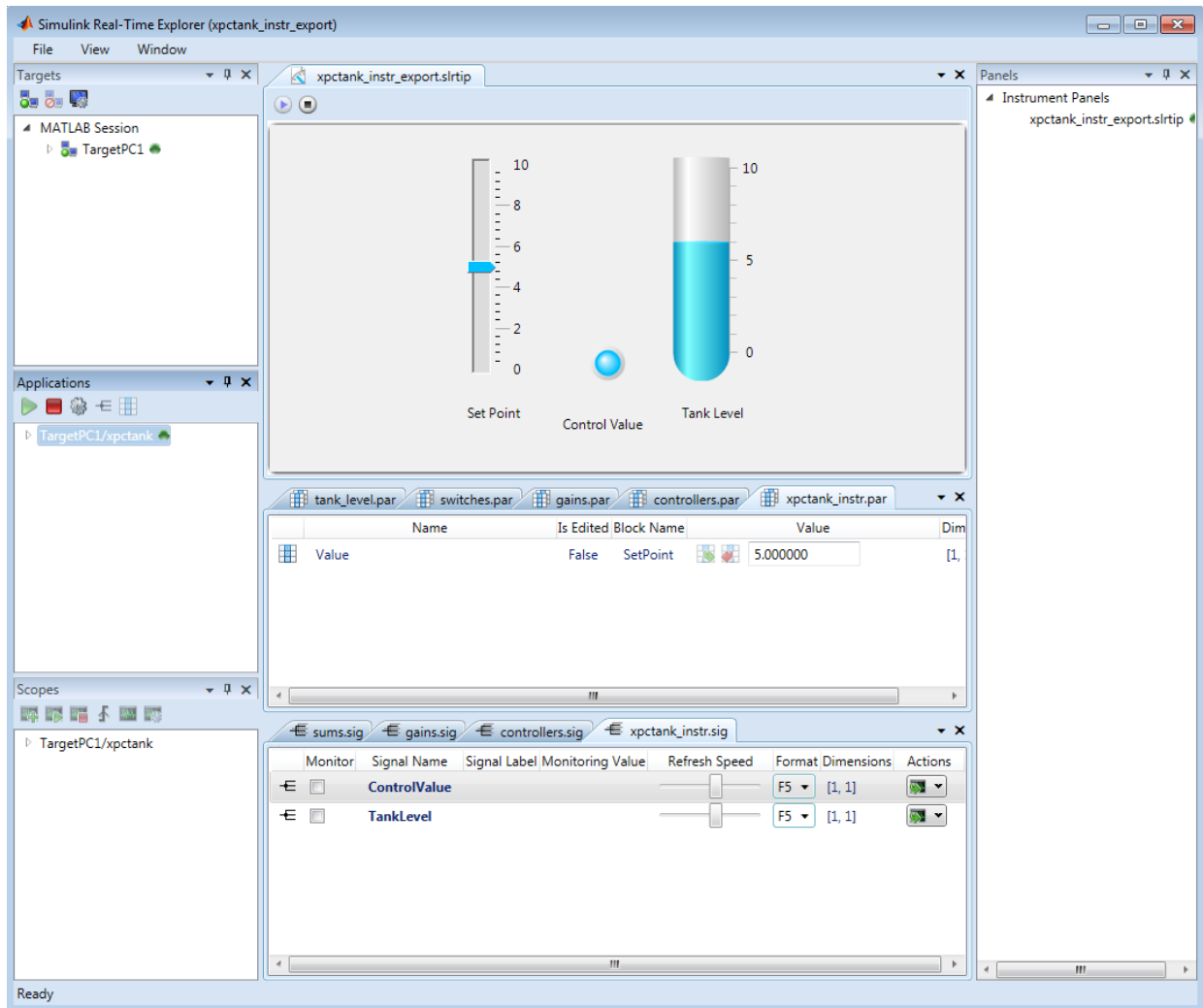
The example uses the instrumented `xpctank` model. Before carrying out this procedure, you must have performed the steps in “Export Explorer Configuration” on page 4-48.

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.

Check 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 4-2

Simulink Real-Time Explorer Instruments

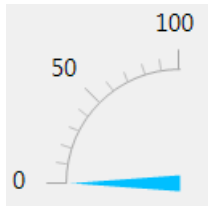
GaugeAngular Properties
GaugeFluidLevel Properties
GaugeLinear
GroupBox
HexadecimalDisplay
Knob
Label
LED
NumericDisplay
NumericEntry
NumericUpDownEntry
Panel
PictureBox
Scope Properties
Slider Properties
SwitchLED


GaugeAngular Properties


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.



To access a dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

Instrument

ScaleDisplay — Selected scale display properties

property group

Open a dialog box to configure the scale display.

Auto — Automatic generator display properties

property group

To set properties of the automatic display generator, click the **Auto** tab.

Mid Included — Insert tick midway between major ticks

False (default) | True

If **MinorCount** is even, space the minor ticks equally around the center tick. If **MinorCount** is odd, replace the center tick with the middle tick.

Fixed Min/Max Majors — Constrain top and bottom tick values

False (default) | True

If **True**, the top and bottom ticks are constrained to be major ticks with min/max values defined by **Min** and **Span**.

Minor Count — Number of minor ticks between major ticks

4 (default) | numeric

Min Text Spacing — Minimum space between scale ticks

1 (default) | numeric

Desired Increment — Display of major tick labels

0 (default) | numeric

The number of labels is $\text{span} / (\text{desired increment} + 1)$. This setting has no effect if the required labels do not fit in the space available in the graphic.

Text Formatting — Text display properties

property group

To set text formatting properties, click the **Text Formatting** tab.

Style — Style of tick display

Number (default) | Thousands | Prefix | Exponent | Price32nds | DateTime | DateTimeUTC

Precision Style — Style of expressing precision

FixedDecimalPoints (default) | SignificantDigits | None

Precision — Number of digits to the right of the decimal point

0 (default) | numeric

Units Text — Text to add to number

text

Open a lines editor and construct text to add to number. By default, the units text appears to the right of the number. If **Style** is **Prefix**, the text appears to the left of the number.

ScaleRange — Selected scale range properties

property group

Specify range of control scale. The properties **Max** and **AngleMax** are calculated from **Min**, **Span**, **AngleMin**, and **AngleSpan**.

Min — Minimum possible value

0 (default) | numeric

Span — Number of values between the minimum and maximum values

100 (default) | numeric

Reverse — Reverse scale direction

False (default) | True

If value is True, reverse the scale so that the values increase in the opposite direction.

Scale Type — Scale type: linear, logarithmic, or split

Linear (default) | Log10 | SplitLinearLog10

Scale type, one of:

- **Linear** — Monotonic increasing or decreasing
- **Log10** — Log base 10 increasing or decreasing
- **SplitLinearLog10** — Monotonic increasing or decreasing in one part, log base 10 in other part

Angle Min — Starting point of scale, in degrees from bottom of circle

180 (default) | numerical

Angle Span — Number of degrees from minimum to full deflection

90 (default) | numerical

Bindings

BindingSrcPath — Name of signal or parameter

text

Specify the hierarchical name of the signal or parameter that is bound to the instrument.

BindingSrcType — Type of object to which the instrument is bound

None (default) | Parameter | Signal

Specify the type of object to which the instrument is bound. The Signal binding is visible only for display instruments.

TargetName — Name of target computer to which the control is connected

text

Bind an instrument instance to a particular target computer. You can bind different instruments in the same instrument panel to different target computers.

See Also

Topics

“Instrumentation for Real-Time Applications” on page 4-2

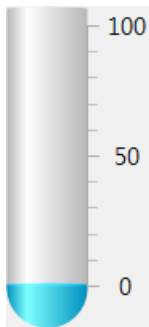
Introduced in R2014a


GaugeFluidLevel Properties


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.



To access a dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

Instrument

ScaleDisplay — Selected scale display properties

property group

Open a dialog box to configure the scale display.

Auto — Automatic generator display properties

property group

To set properties of the automatic display generator, click the **Auto** tab.

Mid Included — Insert tick midway between major ticks

False (default) | True

If **MinorCount** is even, space the minor ticks equally around the center tick. If **MinorCount** is odd, replace the center tick with the middle tick.

Fixed Min/Max Majors — Constrain top and bottom tick values

False (default) | True

If True, the top and bottom ticks are constrained to be major ticks with min/max values defined by **Min** and **Span**.

Minor Count — Number of minor ticks between major ticks

4 (default) | numeric

Min Text Spacing — Minimum space between scale ticks

1 (default) | numeric

Desired Increment — Display of major tick labels

0 (default) | numeric

The number of labels is span / (desired increment + 1). This setting has no effect if the required labels do not fit in the space available in the graphic.

Text Formatting — Text display properties

property group

To set text formatting properties, click the **Text Formatting** tab.

Style — Style of tick display

Number (default) | Thousands | Prefix | Exponent | Price32nds | DateTime | DateTimeUTC

Precision Style — Style of expressing precision

FixedDecimalPoints (default) | SignificantDigits | None

Precision — Number of digits to the right of the decimal point

0 (default) | numeric

Units Text — Text to add to number

text

Open a lines editor and construct text to add to number. By default, the units text appears to the right of the number. If **Style** is **Prefix**, the text appears to the left of the number.

ScaleRange — Selected scale range properties

property group

Specify range of control scale. The property **Max** is calculated from **Min** and **Span**.

Min — Minimum possible value

0 (default) | numeric

Span — Number of values between the minimum and maximum values

100 (default) | numeric

Reverse — Reverse scale direction

False (default) | True

If value is True, reverse the scale so that the values increase in the opposite direction.

Scale Type — Scale type: linear, logarithmic, or split

Linear (default) | Log10 | SplitLinearLog10

Scale type, one of:

- **Linear** — Monotonic increasing or decreasing
- **Log10** — Log base 10 increasing or decreasing
- **SplitLinearLog10** — Monotonic increasing or decreasing in one part, log base 10 in other part

Bindings

BindingSrcPath — Name of signal or parameter

text

Specify the hierarchical name of the signal or parameter that is bound to the instrument.

BindingSrcType — Type of object to which the instrument is bound

None (default) | Parameter | Signal

Specify the type of object to which the instrument is bound. The **Signal** binding is visible only for display instruments.

TargetName — Name of target computer to which the control is connected

text

Bind an instrument instance to a particular target computer. You can bind different instruments in the same instrument panel to different target computers.

See Also

See Also

“Instrument a Tank Model” on page 4-32

Topics

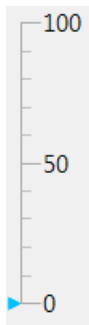
“Instrumentation for Real-Time Applications” on page 4-2

Introduced in R2014a

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 dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

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
DesiredIncrement	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.
FixedMinMaxMajor	If True , the top and bottom ticks are constrained to be major ticks with min/max values defined by Min and Span
MidIncluded	If True , insert a tick halfway between major ticks. If MinorCount is even, space the minor ticks equally around the center tick. If MinorCount is odd, replace the center tick with the middle tick. If
MinorCount	Number of minor ticks between major ticks
MinTextSpacing	Minimum space between scale ticks

Scale Text Display

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

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

UnitsText	Display unit next to tick labels
------------------	----------------------------------

General Scale Range

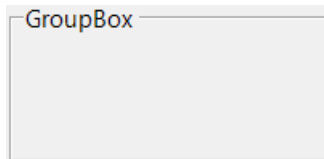
The root node of these parameters is **Instrument+ScaleRange**.

Parameter	Usage
Min	Minimum possible value
Reverse	If True, flip the display to increase in the opposite direction
ScaleType	One of the values Linear , Log10 , and SplitLinearLog10
Span	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
AutoSize	If True , the box expands at design time to make visible the instruments within it
AutoSizeMode	Possible values are GrowAndShrink and GrowOnly . The default is GrowOnly .

HexadecimalDisplay

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 dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

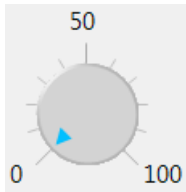
These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

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

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 dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

OffSwitch Graphic Display

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

Parameter	Usage
Enabled	If True, the switch is visible

Parameter	Usage
On	If True, the switch is on

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
DesiredIncrement	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.
FixedMinMaxMajor	If True, the top and bottom ticks are constrained to be major ticks with min/max values defined by Min and Span
MidIncluded	If True, insert a tick halfway between major ticks. If MinorCount is even, space the minor ticks equally around the center tick. If MinorCount is odd, replace the center tick with the middle tick. If
MinorCount	Number of minor ticks between major ticks
MinTextSpacing	Minimum space between scale ticks

Scale Text Display

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

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

General Scale Range

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

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

Angular Scale Range

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

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

Label

Graphic container for text

Description

Label


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
Text	Contains the text displayed by the label
TextAlign	<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 dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

General Parameters

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

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

Indicator Parameters

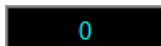
The root node of these parameters is **Instrument+Indicator**.

Parameter	Usage
ColorActive	Indicator color if signal value is 1.
ColorInactive	Indicator color if signal value is 0.

NumericDisplay

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 dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

General Parameters

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

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

Value Display

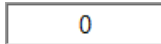
The root node of these parameters is **Iocomp+TextFormatting**.

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

NumericEntry

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 dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

Text Display

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

Parameter	Usage
Precision	Number of digits to the right of the decimal point
PrecisionStyle	One of the values <code>FixedDecimalPoints</code> , <code>SignificantDigits</code> , <code>None</code>

Style	One of the values Number, Thousands, Prefix, Exponent, Price32nds, DateTime, DateTimeUTC
UnitsText	Display unit next to tick labels

NumericUpDownEntry

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 dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

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
DecimalPlaces	Number of decimal places to display
Increment	Value to add or subtract in response to an up-arrow or down-arrow
Maximum	Maximum data value
Minimum	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
AutoScroll	If True , the box scrolls at run time to make fully visible partially visible instruments within it.
AutoSize	If True , the box expands at design time to make visible the instruments within it.
AutoSizeMode	Possible values are GrowAndShrink and GrowOnly . The default is GrowOnly

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 dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

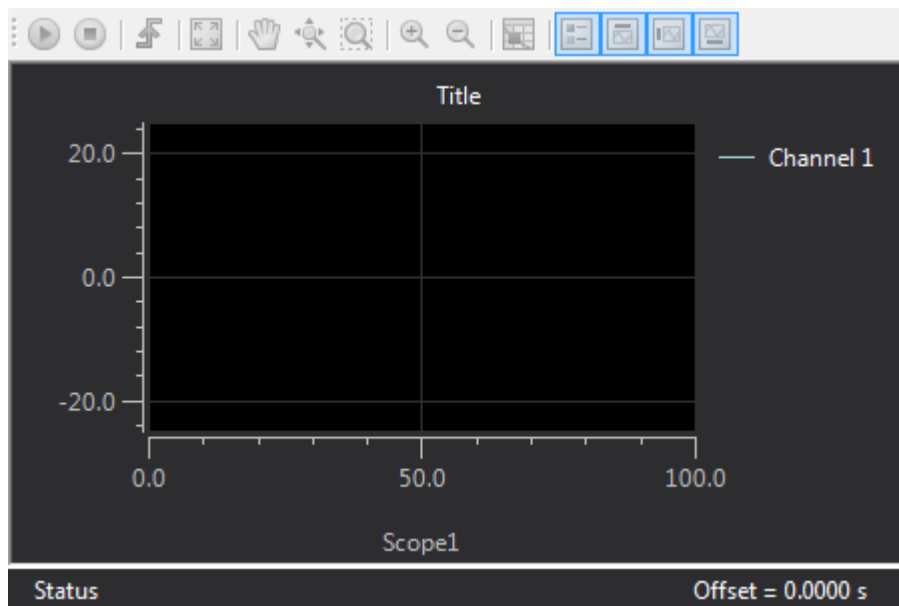
Parameter	Usage
SizeMode	Possible values are Normal, StretchImage, AutoSize, CenterImage, and Zoom. The default is Normal

Scope Properties

Graphic instrument to display waveforms











Description


Use the Scope instrument to display time-varying real-valued data, such as electronic waveforms.




The block provides controls that you can access at run time.

Action	Icon	Purpose
Start Acquisition		Enables the scope to trigger.
Stop Acquisition		Prevents the scope from triggering.
Force trigger		Triggers an acquisition manually.
AutoScale		Rescales the scope axes to the bounds of each sample.

Action	Icon	Purpose
Axis Scroll		Enables you to drag the visible portion of the axes left, right, up, and down.
Axis Zoom		Enables you to stretch the axes left, right, up, and down.
Zoom Box		Enables you to select a box into which the display zooms.
Zoom In		Zooms the display inward from the current center.
Zoom Out		Zooms the display outward from the current center.
Data-Cursor		Creates a cursor that shows value and time data.
Show Legend		Displays the signal names and the colors of the signal traces in the top, right corner of the scope display.
Show Label		Displays the scope label in the top center of the scope display.
Show Y-Axis		Displays the Y-axis to the left of the scope display.
Show X-Axis		Displays the X-axis on the bottom of the scope display.

To access a dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

Instrument

NumSamples — Number of contiguous samples that the scope displays

500 (default) | integer

The number of samples determines how much data the scope displays at a time. The larger this value, the longer the scope takes to refresh.

PreTriggerViewing — Percentage of samples that the scope captures before a trigger event

0 (default) | integer

This property determines the horizontal (time axis) position of the trigger event in the display.

TriggerLevel — Level at which a signal trigger becomes active

0 (default) | integer

When the trigger signal crosses this value in the direction that you specified in **TriggerSlope**, the trigger becomes active.

Dependency

This property takes effect only if **TriggerSource** is SIGNAL.

TriggerMode — Scope update policy

SINGLESHOT (default) | REPEATED

Specify how often the display is updated:

- **SINGLESHOT** — Trigger once and stops the scope. To capture again, manually restart the scope.
- **REPEATED** — Trigger every time the trigger condition is satisfied. To stop the display, manually stop the scope.

TriggerScope — Name of triggering scope

text

Specify the scope from which this instance of the scope instrument triggers.

Dependency

This property takes effect only if **TriggerSource** is SCOPE.

TriggerSignal — Name of triggering signal

text

Specify the signal from which this instance of the scope instrument triggers.

Dependency

This property takes effect only if **TriggerSource** is SIGNAL.

TriggerSlope — Slope of signal waveform that triggers this scope

EITHER (default) | RISING | FALLING

Specify the crossing direction at which the trigger becomes active:

- **EITHER** — Trigger when the signal rises and when it falls.
- **RISING** — Trigger only when the signal rises.
- **FALLING** — Trigger only when the signal falls.

Specify the crossing value with property **TriggerLevel**.


Dependency

This property takes effect only if **TriggerSource** is SIGNAL.

TriggerSource — Source of trigger event


FREERUN (default) | MANUAL | SIGNAL | SCOPE

Specify the trigger condition:

- **FREERUN** — Display triggers at every sample time.
- **MANUAL** — Display triggers when you click the **Force trigger** button .
- **SIGNAL** — Display triggers when a specific signal crosses a specific value in a rising or falling direction.
- **SCOPE** — Display triggers when a specified scope is triggered.

Dependency

This parameter has the following dependencies:

- When **TriggerSource** is MANUAL, the **Force trigger** button  becomes available.
- When **TriggerSource** is SIGNAL, properties **TriggerSignal**, **TriggerSlope**, and **TriggerLevel** take effect.

- When **TriggerSource** is SCOPE, property **TriggerScope** takes effect.

YAxesLimits — Positive and negative limits of Y-axis

max-min pair

Specify **Max** and **Min** limits to the Y-axis of the display.

Bindings

Signals — List of signals that are connected to scope instrument

list of signal names

Enter signal names by using the Signal Collection Editor, which contains a **Members** list and a **Properties** list.

Example: SetPoint, ControlValue, TankLevel

TargetName — Name of target computer to which the control is connected

text

Bind an instrument instance to a particular target computer. You can bind different instruments in the same instrument panel to different target computers.

See Also

Topics

“View Signal Waveforms with Scope Instrument” on page 4-19

“View Signal Waveforms with Signal Triggered Scope Instrument” on page 4-27

“Triggering Scope Instruments”

“Instrumentation for Real-Time Applications” on page 4-2

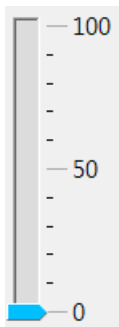
Introduced in R2017a


Slider Properties


Graphic instrument to set parameter values

Description

Use the **Slider** instrument to set the approximate value of real-valued data, such as voltage or current.



To access a dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

Instrument

ScaleDisplay — Selected scale display properties

property group

Open a dialog box to configure the scale display.

Auto — Automatic generator display properties

property group

To set properties of the automatic display generator, click the **Auto** tab.

Mid Included — Insert tick midway between major ticks

False (default) | True

If **MinorCount** is even, space the minor ticks equally around the center tick. If **MinorCount** is odd, replace the center tick with the middle tick.

Fixed Min/Max Majors — Constrain top and bottom tick values

False (default) | True

If True, the top and bottom ticks are constrained to be major ticks with min/max values defined by **Min** and **Span**.

Minor Count — Number of minor ticks between major ticks

4 (default) | numeric

Min Text Spacing — Minimum space between scale ticks

1 (default) | numeric

Desired Increment — Display of major tick labels

0 (default) | numeric

The number of labels is $\text{span} / (\text{desired increment} + 1)$. This setting has no effect if the required labels do not fit in the space available in the graphic.

Text Formatting — Text display properties

property group

To set text formatting properties, click the **Text Formatting** tab.

Style — Style of tick display

Number (default) | Thousands | Prefix | Exponent | Price32nds | DateTime | DateTimeUTC

Precision Style — Style of expressing precision

FixedDecimalPoints (default) | SignificantDigits | None

Precision — Number of digits to the right of the decimal point

0 (default) | numeric

Units Text — Text to add to number

text

Open a lines editor and construct text to add to number. By default, the units text appears to the right of the number. If **Style** is **Prefix**, the text appears to the left of the number.

ScaleRange — Selected scale range properties

property group

Specify range of control scale. The property **Max** is calculated from **Min** and **Span**.

Min — Minimum possible value

0 (default) | numeric

Span — Number of values between the minimum and maximum values

100 (default) | numeric

Reverse — Reverse scale direction

False (default) | True

If value is True, reverse the scale so that the values increase in the opposite direction.

Scale Type — Scale type: linear, logarithmic, or split

Linear (default) | Log10 | SplitLinearLog10

Scale type, one of:

- **Linear** — Monotonic increasing or decreasing
- **Log10** — Log base 10 increasing or decreasing
- **SplitLinearLog10** — Monotonic increasing or decreasing in one part, log base 10 in other part

Bindings

BindingSrcPath — Name of signal or parameter

text

Specify the hierarchical name of the signal or parameter that is bound to the instrument.

BindingSrcType — Type of object to which the instrument is bound

None (default) | Parameter | Signal

Specify the type of object to which the instrument is bound. The **Signal** binding is visible only for display instruments.

TargetName — Name of target computer to which the control is connected

text

Bind an instrument instance to a particular target computer. You can bind different instruments in the same instrument panel to different target computers.

See Also

Topics

“View Signal Waveforms with Scope Instrument” on page 4-19

“View Signal Waveforms with Signal Triggered Scope Instrument” on page 4-27

“Instrument a Tank Model” on page 4-32

“Triggering Scope Instruments”

“Instrumentation for Real-Time Applications” on page 4-2

Introduced in R2014a

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 dialog box containing key instrument properties, select the instrument and in the top, right corner of the instrument, click the **Tasks** button .

To access the complete **Properties** list for an instrument, select the instrument in the instrument panel. To access a dialog box for a property group, click the group. To the right of the group, click the continuation button .

These properties are a selection from the available properties. For more properties, see “Graphical Properties” on page 4-10.

General Parameters

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

Parameter	Usage
AutoSize	If True, 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.
ColorInactive	Indicator color if signal value is 0.

Target Computer Command-Line Interface Reference

Target Computer Commands

You can interact with the real-time application after it has been loaded to the target computer by using a limited set of target computer commands.

- Function commands — Interact directly with the scope or target.
- Property commands — Work with target and scope properties.
- Variable commands — Alias target computer command-line interface commands with 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.

To read the target computer console log, call `SimulinkRealTime.utils.getConsoleLog`.

In this section...

“Target Object Function Commands” on page 6-2

“Target Object Property Commands” on page 6-3

“Scope and Video Object Function Commands” on page 6-4

“Scope Object Property Commands” on page 6-6

“Aliasing with Variable Commands” on page 6-10

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.

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.

To read signal and parameter and signal values, use signal and parameter names (for example, `S1`, `P1`). To both read and write parameter values, use parameter indexes (for example, `0`, `1`).

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

Target Computer Command	Description	MATLAB Equivalent
P#	Display the value of the block parameter with index #. For example, P2 displays the value of block parameter 2.	getparam(tg, param_index)
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. The functions `addscope` and `remscope` are target object functions that also run on the development computer.

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.

To add and remove scopes, use scope numbers (for example, 0, 1) and not scope names (for example, `Scope 1`, `Scope 2`). To add and remove signals from scopes, use signal indexes (for example, 0, 1) and not signal names (for example, `S0`, `S1`).

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>

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

Target Computer Command	Description	MATLAB Equivalent
show Video video_index	With value video_index, expand video display video_index from an icon.	
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 the properties shown in the following table.

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.

To change scope properties, use scope indexes (for example, 0, 1) and not scope names (for example, Scope 1, Scope 2).

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

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'

Target Computer Command	Description	MATLAB Equivalent
scopemode scope_index = 0	With value 0 or numerical, scope scope_index displays signal values as text.	sc.DisplayMode = 'numerical'
scopemode scope_index = numerical	With value 1 or redraw, scope scope_index plots signal values when numsamples samples have been acquired.	sc.DisplayMode = 'redraw'
scopemode scope_index = 1	With value 3 or rolling, scope scope_index plots signal values at every sample time.	sc.DisplayMode = 'rolling'
scopemode scope_index = redraw	Value 2, sliding, will be removed in a future release. It behaves like value 3, rolling.	
scopemode scope_index = 3		
scopemode scope_index = rolling		

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

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>ylimit 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>ylimit scope_index = auto</code>	With value <code>auto</code> , the values being displayed determine the lower and upper <i>y</i> -axis values.	<code>sc.YLimit = 'auto'</code>

Aliasing with Variable Commands

You can set a variable to a command character vector, 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 character vector.
<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.

See Also

`SimulinkRealTime.utils.getConsoleLog`

Simulink Real-Time Performance Advisor Checks

- “Simulink Real-Time Performance Advisor Checks” on page 7-2
- “Baseline” on page 7-4
- “System Target File Compatibility” on page 7-5
- “Profiling Settings” on page 7-6
- “Check Target” on page 7-7
- “Real-Time Performance Baseline” on page 7-8
- “Determine minimum sample time” on page 7-9
- “Real-Time” on page 7-10
- “Outport Logging” on page 7-11
- “EtherCAT Synchronous SDO” on page 7-12
- “Concurrent execution” on page 7-13
- “Final Validation” on page 7-14

Simulink Real-Time Performance Advisor Checks

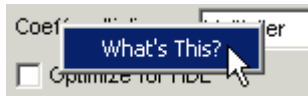
Parameter	Description
“Baseline” on page 7-4	Checks preconditions for real-time execution, and then measures its initial performance to establish a baseline.
“System Target File Compatibility” on page 7-5	Checks that the system target file is compatible with the real-time advisor workflow.
“Profiling Settings” on page 7-6	Checks that the execution profiling settings are compatible with the real-time advisor workflow.
“Check Target” on page 7-7	Pings the target computer and verifies that it is in a clean state.
“Real-Time Performance Baseline” on page 7-8	Collects execution-time measurements and establishes a performance baseline.
“Determine minimum sample time” on page 7-9	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.
“Real-Time” on page 7-10	Checks the real-time application and application setup, and recommends changes to improve performance.
“Outport Logging” on page 7-11	Checks for the use of Outport blocks for data logging. Data logging using Outport blocks can slow down execution.
“EtherCAT Synchronous SDO” on page 7-12	Checks for the use of EtherCAT Synchronous SDO blocks. EtherCAT Synchronous SDO blocks can slow down execution.
“Concurrent execution” on page 7-13	Checks if you can perform concurrent execution of the real-time application on a multicore target computer.

Parameter	Description
“Final Validation” on page 7-14	Validates the overall performance improvement that your changes make in real-time execution time and accuracy.

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.

To get help on an option

- 1 Right-click the option text label.
- 2 From the context menu, select **What's This**.



See Also

- “Improve Performance of Multirate Model”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

Baseline

Checks preconditions for real-time execution, and then measures its initial performance to establish a baseline.

Performance Advisor later uses this baseline to compare performance before and after you implement the improvements that Performance Advisor recommends.

See Also

- “Generate Baseline”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

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 `slrt.tlc`.

See Also

- “Generate Baseline”
- “Set Configuration Parameters”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

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

- “Generate Baseline”
- “Configure Real-Time Application for Profiling”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

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

- “Generate Baseline”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

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

- “Generate Baseline”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

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

- “Generate Baseline”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

Real-Time

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

See Also

- “Perform Real-Time Checks”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

Output Logging

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

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

See Also

- “Perform Real-Time Checks”
- Scope
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

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

- “Perform Real-Time Checks”
- “EtherCAT”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

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

- “Perform Real-Time Checks”
- “Concurrent Execution with Multicore Target Computer”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)

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

- “Final Validation”
- “Performance Optimization”
- “Improve Simulation Performance Using Performance Advisor” (Simulink)
- “Comparing Performance” (Simulink)