

Simulink® Desktop Real-Time™

User's Guide



MATLAB® & SIMULINK®

R2022a



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.
1 Apple Hill Drive
Natick, MA 01760-2098

Simulink® Desktop Real-Time™ User's Guide

© COPYRIGHT 1999–2022 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

January 1999	First printing	New for Version 1.0 (Release 11.0)
January 2000	Second printing	Revised for Version 1.5 (Release 11.1+)
September 2000	Third printing	Revised for Version 2.0 (Release R12)
June 2001	Online only	Revised for Version 2.1 (Release R12.1)
July 2002	Online only	Revised for Version 2.2 (Release 13)
June 2004	Fourth printing	Revised for Version 2.5 (Release 14)
October 2004	Fifth printing	Revised for Version 2.5.1 (Release 14SP1)
March 2005	Online only	Revised for Version 2.5.2 (Release 14SP2)
September 2005	Online only	Revised for Version 2.6 (Release 14SP3)
March 2006	Online only	Revised for Version 2.6.1 (Release 2006a)
September 2006	Online only	Revised for Version 2.6.2 (Release 2006b)
March 2007	Online only	Revised for Version 2.7 (Release 2007a)
September 2007	Online only	Revised for Version 3.0 (Release 2007b)
March 2008	Online only	Revised for Version 3.1 (Release 2008a)
October 2008	Online only	Revised for Version 3.2 (Release 2008b)
March 2009	Online only	Revised for Version 3.3 (Release 2009a)
September 2009	Online only	Revised for Version 3.4 (Release 2009b)
March 2010	Online only	Revised for Version 3.5 (Release 2010a)
September 2010	Online only	Revised for Version 3.6 (Release 2010b)
April 2011	Online only	Revised for Version 3.7 (Release 2011a)
September 2011	Online only	Revised for Version 3.8 (Release 2011b)
March 2012	Online only	Revised for Version 4.0 (Release 2012a)
September 2012	Online only	Revised for Version 4.1 (Release 2012b)
March 2013	Online only	Revised for Version 4.2 (Release 2013a)
September 2013	Online only	Revised for Version 4.3 (Release 2013b)
March 2014	Online only	Revised for Version 4.4 (Release 2014a)
October 2014	Online only	Revised for Version 4.5 (Release 2014b)
March 2015	Online only	Revised for Version 5.0 (Release 2015a)
September 2015	Online only	Revised for Version 5.1 (Release 2015b)
March 2016	Online only	Revised for Version 5.2 (Release 2016a)
September 2016	Online only	Revised for Version 5.3 (Release 2016b)
March 2017	Online only	Revised for Version 5.4 (Release 2017a)
September 2017	Online only	Revised for Version 5.5 (Release 2017b)
March 2018	Online only	Revised for Version 5.6 (Release 2018a)
September 2018	Online only	Revised for Version 5.7 (Release 2018b)
March 2019	Online only	Revised for Version 5.8 (Release 2019a)
September 2019	Online only	Revised for Version 5.9 (Release 2019b)
March 2020	Online only	Revised for Version 5.10 (Release 2020a)
September 2020	Online only	Revised for Version 5.11 (Release 2020b)
March 2021	Online only	Revised for Version 5.12 (Release 2021a)
September 2021	Online only	Revised for Version 5.13 (Release 2021b)
March 2022	Online only	Revised for Version 5.14 (Release 2022a)

Getting Started

1

Simulink Desktop Real-Time Product Description	1-2
Key Features	1-2
Real-Time Execution in Connected IO Mode	1-3
Real-Time Execution in Run in Kernel Mode	1-5

Installation and Configuration

2

Software Components	2-2
MATLAB Environment	2-2
Simulink Software	2-2
Simulink Coder Software	2-2
Known Limitations	2-3
Install Real-Time Kernel	2-4
Install the Kernel Using MATLAB	2-4
Uninstall the Kernel	2-5
Run Confidence Test	2-6
Run the Model sldrtex_vdp	2-6
Display Status Information	2-8
Examples Library	2-9

Basic Procedures

3

Prepare for Real-Time Execution	3-2
Prepare I/O Devices	3-2
Prepare Real-Time Application	3-2
Create a Real-Time Application	3-4
Create a Simulink Model	3-5
File System I/O	3-9

Configure a Model for Simulink Desktop Real-Time	3-10
Specify a Default Configuration Set	3-10
Enter Configuration Parameters Manually	3-11
Enter Scope Parameters for Signal Tracing	3-12
Simulate Model in Connected IO Mode	3-15
Set Run in Kernel Mode Code Generation Parameters	3-17
Prepare Run in Kernel Mode Application	3-20
Set Run in Kernel Mode (External Mode) Scope Parameters	3-21
Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time	3-24
Execute Real-Time Application in Run in Kernel Mode by Using Step by Step Commands	3-26
Execute Real-Time Application with S-Functions in Run in Kernel Mode	3-28
Run Application from MATLAB Command Line	3-30
Connected IO Mode (Normal Mode)	3-30
Accelerator Mode	3-30
Run in Kernel Mode (External Mode)	3-30
Inspect Simulink® Desktop Real-Time™ Signals with Simulation Data Inspector	3-32
Signal Logging to the Workspace	3-35
Set Scope Parameters for Logging to Workspace	3-36
Set Run in Kernel Mode Properties for Logging to Workspace	3-38
Plot Signal Data Logged to Workspace	3-40
Signal Logging to a File	3-42
Set Scope Parameters for Logging to File	3-43
Set Run in Kernel Mode Properties for Logging to File	3-45
Set Run in Kernel Mode Data Archiving Parameters	3-47
Plot Signal Data Logged to File	3-49
Tunable Block Parameters and Tunable Global Parameters	3-51
Tunable Parameters	3-51
Inlined Parameters	3-51
Tune Parameters by Using Run in Kernel Mode	3-52
Tune Parameters by Using Hold Updates and Update All Parameters ...	3-52
Tune Parameters by Using the MATLAB Language	3-52

Tune Block Parameters by Using the Block Dialog Box	3-54
Tune Block Parameters with Data Navigation	3-57
Create Parameter Object	3-57
Tune Parameter Object	3-59
Sweep MATLAB Variables with MATLAB Scripting	3-62

Boards, Blocks, and Drivers

4

Use I/O Boards	4-2
Install and Configure I/O Boards and Drivers	4-2
PCI Bus Board	4-3
Compact PCI Board	4-4
PCMCIA Board	4-4
Use I/O Driver Blocks	4-5
View Simulink Desktop Real-Time Library	4-5
Route Signals from an I/O Block	4-5
Configure Channel Selection	4-6
Use Analog I/O Drivers	4-9
Configure I/O Driver Characteristics	4-9
Normalize Scaling for Analog Inputs	4-9
Use Vector CAN Drivers	4-12
Use Video Input Driver (Mac OS)	4-13

Custom I/O Driver Blocks

A

Custom I/O Driver Basics	A-2
Supported C Functions	A-2
Unsupported C Functions	A-2
Incompatibility with Operating System API Calls	A-3
I/O Register Access from S-Functions Limitation	A-3

Simulink Desktop Real-Time Examples

5

Real-Time Van der Pol Simulation	5-2
Water Tank Model with Dashboard	5-5

Real-Time Signal Generator	5-8
Real-Time Controller	5-10
Real-Time Filter	5-12
Frequency Measurement	5-14
PWM Frequency and Duty Measurement	5-16
Packet Input/Output	5-18
Stream Input/Output	5-21
Video Input	5-24
CAN Input/Output	5-26
CAN Input/Output with Vehicle Network Toolbox	5-29
Execution Time Measurement and Block Profiling	5-32
Apply Simulink Desktop Real-Time Model Templates to Create Real-Time Models	5-36
UDP String Data and Message Handling	5-38
Sync Models by Using Arduino Connected I/O Board	5-40

Troubleshooting

6

Troubleshoot Missing Desktop Real-Time Tab	6-2
What This Issue Means	6-2
Try This Workaround	6-2
Troubleshoot sldrtxt Incorrect Version Error	6-3
What This Issue Means	6-3
Try This Workaround	6-3
Troubleshoot Delayed or Missing Scope Output	6-4
What This Issue Means	6-4
Try This Workaround	6-4
Troubleshoot Signals Not Plotted in Scope Blocks	6-5
What This Issue Means	6-5
Try This Workaround	6-5
Troubleshoot Vendor Software Missing Issues	6-7
What This Issue Means	6-7
Try This Workaround	6-7

Troubleshoot Builds of Referenced Models	6-8
What This Issue Means	6-8
Try This Workaround	6-8
Troubleshoot Slow or Halted Simulation on Windows	6-9
What This Issue Means	6-9
Try This Workaround	6-9
Troubleshoot C++ Standard Template Library (STL) Compilation Errors for Real-Time Application	6-12
What This Issue Means	6-12
Try This Workaround	6-12

Getting Started

- “Simulink Desktop Real-Time Product Description” on page 1-2
- “Real-Time Execution in Connected IO Mode” on page 1-3
- “Real-Time Execution in Run in Kernel Mode” on page 1-5

Simulink Desktop Real-Time Product Description

Run Simulink models in real time on your computer

Simulink Desktop Real-Time provides a real-time kernel for executing Simulink models on a Windows® or Mac laptop or desktop. It includes library blocks that connect to a range of I/O devices. You can create and tune a real-time system for rapid prototyping or hardware-in-the-loop simulation with your computer.

Simulink Desktop Real-Time supports real-time performance with lower sample rate for **Connected IO** mode simulation in Simulink, and supports higher sample rates for **Run in Kernel** mode simulation in Simulink with Simulink Coder™.

Key Features

- Real-time closed-loop execution of Simulink models
- Signal visualization and parameter tuning while model is running
- Real-time performance approaching a 1 kHz sample rate in Simulink **Connected IO** mode
- Real-time performance approaching a 20 kHz sample rate in Simulink **Run in Kernel** mode (with Simulink Coder)
- Blocks supporting more than 250 I/O devices (including analog I/O, digital I/O, counters, encoders, and frequency output) and communication protocols (including UDP, serial, and CAN)
- Connection to I/O devices installed in your computer or in a Thunderbolt expansion chassis

Real-Time Execution in Connected IO Mode

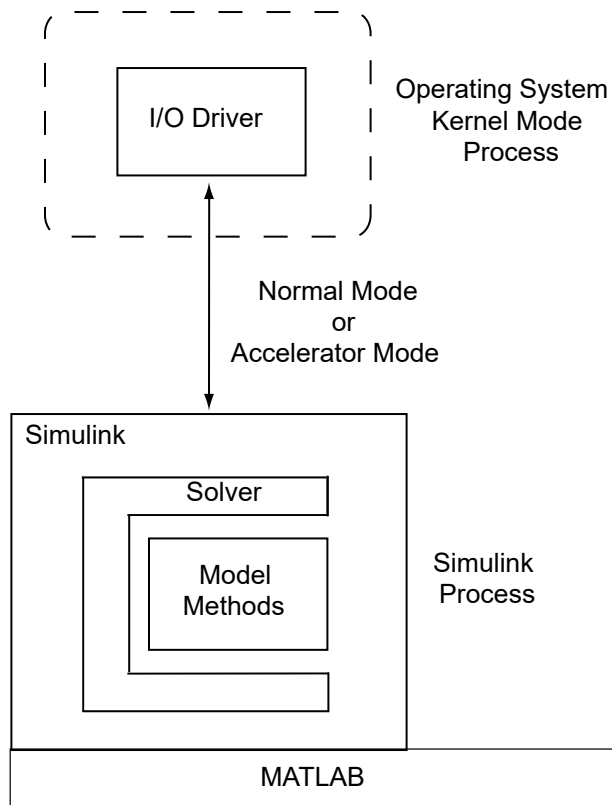
Simulink Desktop Real-Time **Connected IO** mode extends normal mode or accelerator mode to run in real time.

The simulation algorithm for a non-real-time normal mode or accelerator model runs entirely within Simulink. The model can use either a fixed-step or a variable-step solver and runs as fast as it can, given the presence of competing operating system processes. It is not synchronized with a real-time clock, and you cannot easily use the model to operate real-time hardware.

You can synchronize a Simulink model with a real-time clock by using Simulink Desktop Real-Time I/O blocks. In **Connected IO** mode, Simulink executes the simulation algorithm, while a separate operating system kernel mode process runs I/O drivers for the I/O blocks. The Simulink process and the kernel mode process run on the host machine by using a shared memory interface to transfer parameter data.

- Signal acquisition — You can capture and display signals from your real-time application while it is running. Simulink retrieves signal data from the I/O driver and displays it in the same Scope blocks that you used for simulating your model in nonreal time.
- Parameter tuning — You can change parameters in your Simulink block diagram and have the new parameters take effect in your Simulink model in real time. The effects then propagate through the I/O driver to the hardware.

Note You cannot run a Simulink Desktop Real-Time model in rapid accelerator mode.



Because only the I/O drivers are synchronized with the real-time clock, Simulink can use either a fixed-step or a variable-step solver. The **Sample Time** setting in the Simulink Desktop Real-Time block does not change the step size of the simulation. For a fixed-step simulation, in the Configuration Parameters, you set the step size in the **Fixed step size** box. For a variable-step simulation, you set the step size by using the **Min Step Size** attribute, or Simulink determines the step size automatically.

In **Connected IO** mode, at each sample interval, Simulink evaluates each real-time block. Simulink writes the input data into a buffer that it passes to the kernel mode process. The kernel mode process propagates the data to the hardware, which writes response data into another buffer. At the next time tick, Simulink reads the response data and propagates it to the rest of the model.

A consequence of this kind of limited synchronization is that your simulation can be configured to miss real-time clock ticks and their associated data points. Ticks can be missed under the following circumstances:

- **Complexity of Model** — The model can be so complex that Simulink cannot keep up with the real-time kernel. In this case, the number of missed ticks increases steadily with time. Once the number of missed ticks exceeds **Maximum Missed Ticks**, an error occurs, even if **Maximum Missed Ticks** is set to a large value. You can identify this situation by a rising straight line on a Scope connected to the optional **Missed Ticks** port.
- **Process Contention** — The model generally executes faster than required to keep up with the kernel. Process contention or some random operating system condition prevents Simulink from executing the model over some time period. In this case, the number of missed ticks jumps to some number, then decreases to zero as Simulink catches up with the kernel. You can identify this situation by a sawtooth-like shape on a Scope connected to the **Missed Ticks** port.
- **Variable-Step Solver** — If you are using a variable-step solver, the number of ticks for each algorithm step can vary during simulation. If Simulink execution does not reach the Simulink Desktop Real-Time blocks in time to synchronize with the tick, the number of missed ticks jumps to some number. As Simulink catches up with the kernel, the number of missed ticks decreases to zero. You can identify this situation by a sawtooth-like shape on a Scope connected to the **Missed Ticks** port.

See Also

Related Examples

- “Simulate Model in Connected IO Mode” on page 3-15
- “Run Application from MATLAB Command Line” on page 3-30
- “Real-Time Van der Pol Simulation” on page 5-2

Real-Time Execution in Run in Kernel Mode

The **Run in Kernel** mode is a higher-performance alternative to “Real-Time Execution in Connected IO Mode” on page 1-3. In **Run in Kernel** mode, you use Simulink Coder to link generated algorithm code to I/O driver code generated from the I/O blocks. The resulting executable runs in operating system kernel mode on the development computer. The executable exchanges parameter data with Simulink through a shared memory interface.

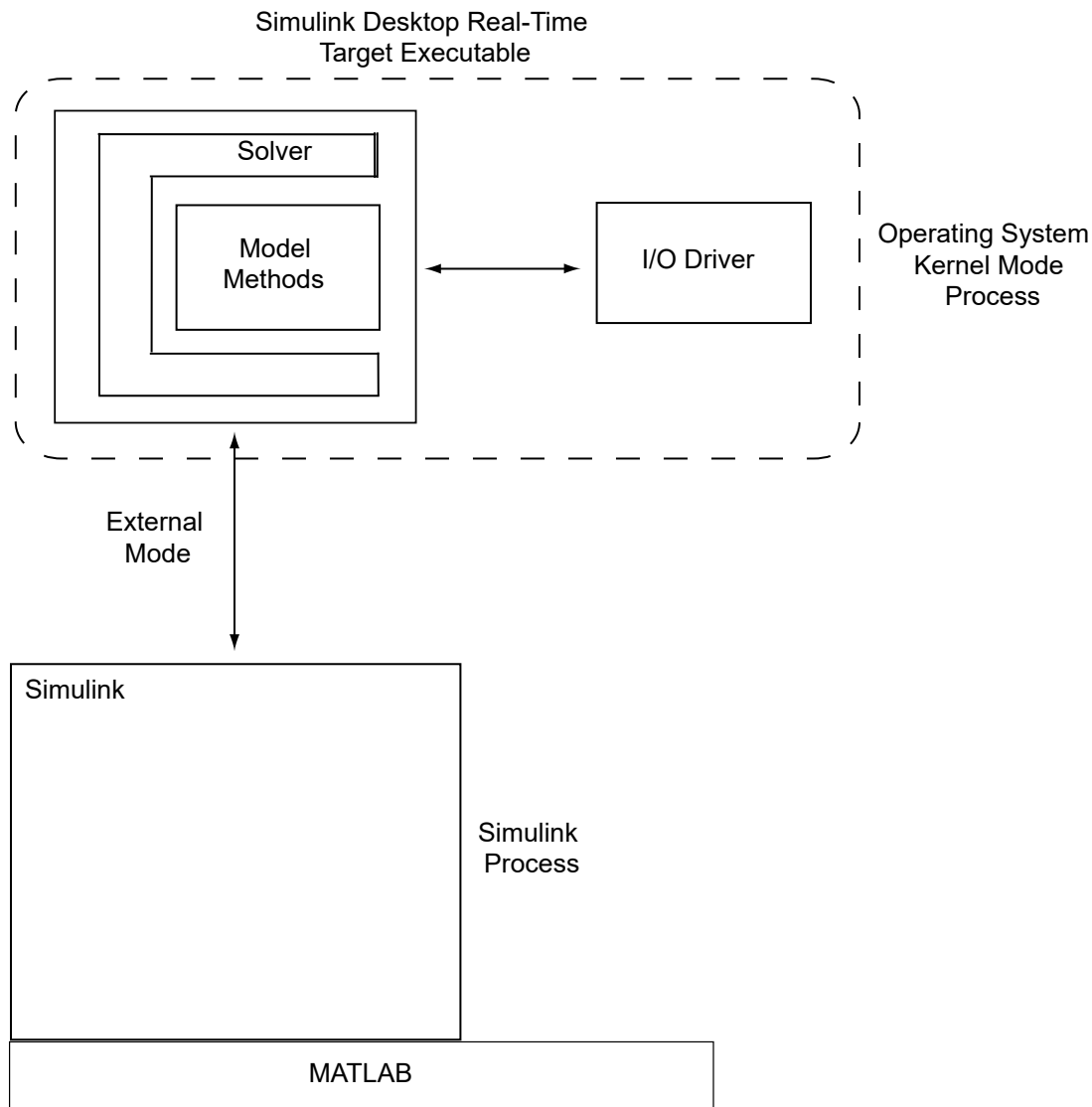
- Signal acquisition — You can capture and display signals from your real-time application while it is running. Signal data is retrieved from the real-time application and displayed in the same Simulink Scope blocks that you used for simulating your model.
- Parameter tuning — You can change parameters in your Simulink block diagram and have the new parameters passed automatically to the real-time application. External mode changes parameters in your real-time application while it is running in real time.

The **Run in Kernel** mode executable is fully synchronized with the real-time clock. The main role of Simulink is to read and display simulation results returned from the executable.

Procedures related to real-time execution in run in kernel mode include:

- 1 “Set Run in Kernel Mode Code Generation Parameters” on page 3-17
- 2 “Prepare Run in Kernel Mode Application” on page 3-20
- 3 “Set Run in Kernel Mode (External Mode) Scope Parameters” on page 3-21
- 4 “Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time” on page 3-24
- 5 “Execute Real-Time Application in Run in Kernel Mode by Using Step by Step Commands” on page 3-26
- 6 “Execute Real-Time Application with S-Functions in Run in Kernel Mode” on page 3-28

Note Use a fixed-step solver in **Run in Kernel** mode.



In **Run in Kernel** mode, the real-time application and the I/O drivers run in the kernel mode process. Using the I/O drivers to communicate with the hardware, the application stores contiguous response data in memory accessible to Simulink until a data buffer is filled. When the buffer is filled, the real-time application continues to run while Simulink transfers the data to the MATLAB® environment through Simulink external mode. Transfer of data is less critical than maintaining deterministic real-time updates at the required sample interval. After model computations are finished, data transfer runs at a lower priority while the process waits for another interrupt to trigger the next model update.

Data captured within one buffer is contiguous. When a buffer of data has been transferred, it is immediately plotted in a Simulink Scope block. You can save the data directly to a MAT-file by using data archiving in Simulink external mode.

With data archiving, you can save each buffer of data to its own MAT-file. The MAT-file names can be automatically incremented, enabling you to capture and store many data buffers. Although points within a buffer are contiguous, the time required to transfer data back to Simulink pauses data

collection until the entire buffer has been transferred. This pause can result in lost sample points between data buffers.

See Also

Related Examples

- “Set Run in Kernel Mode Code Generation Parameters” on page 3-17
- “Prepare Run in Kernel Mode Application” on page 3-20
- “Set Run in Kernel Mode (External Mode) Scope Parameters” on page 3-21
- “Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time” on page 3-24
- “Run Application from MATLAB Command Line” on page 3-30
- “Real-Time Van der Pol Simulation” on page 5-2

Installation and Configuration

- “Software Components” on page 2-2
- “Install Real-Time Kernel” on page 2-4
- “Run Confidence Test” on page 2-6

Software Components

Simulink Desktop Real-Time is a self-targeting rapid prototyping system where the host and the target computer are the same computer.

MATLAB Environment

The MATLAB environment provides the design and data analysis tools that you use when creating and testing Simulink models. In particular, see:

- “Importing and Exporting Data”
- “Plotting Data”

Simulink Software

Simulink software provides an environment where you model your physical system and controller as a block diagram. You create the block diagram by using a mouse to connect blocks and a keyboard to edit block parameters. You can use Simulink Desktop Real-Time software with most Simulink blocks, including discrete-time and continuous-time systems. Simulink Coder supports C code S-functions.

With Simulink Desktop Real-Time software, you can remove the physical system model and replace it with Simulink Desktop Real-Time I/O driver blocks connected to your sensors and actuators. The Simulink Desktop Real-Time I/O library supports more than 200 boards.

Note For some boards, the Simulink Desktop Real-Time software does not support some of the board functions. Check the MathWorks® website for an updated list of supported boards and functions at www.mathworks.com/hardware-support/simulink-desktop-real-time.html.

Simulink Coder Software

The Simulink Coder software provides utilities to convert your Simulink models into C code and then compile the code into a real-time executable.

Note

- Simulink Coder is required for **Run in Kernel** mode.
 - Compiler support is included as part of the product installation. No additional or external compiler is required.
 - MATLAB Coder is required for Simulink Coder installation.
-

Simulink Desktop Real-Time software is designed for maximum flexibility during rapid prototyping. This flexibility allows parameter tuning and signal tracing during a real-time run, but increases the size of the generated code. However, Simulink Coder code generation software provides other code formats that generate more compact code for embedded applications.

Known Limitations

- In **Run in Kernel** mode, the Simulink Desktop Real-Time software does not support the following:
 - Blocks that do not support code generation
 - To File blocks
- Limitations with Simulink Coder code generation software:
 - In **Run in Kernel** mode, MATLAB S-functions have limited support. See “Execute Real-Time Application with S-Functions in Run in Kernel Mode” on page 3-28.
 - When you use a continuous-time system and generate code for **Run in Kernel** mode execution with Simulink Coder code generation software, you must use a fixed-step integration algorithm.
 - The Simulink Coder product provides an API for the MATLAB Parallel Server™ or Parallel Computing Toolbox™ products to perform parallel builds that reduce build time for referenced models. However, this API does not support parallel builds for models whose system target file parameter is set to `sldrt.tlc` or `sldrtert.tlc`. In other words, you cannot perform parallel builds for Simulink Desktop Real-Time.

See Also

More About

- “Importing and Exporting Data”
- “Plotting Data”

External Websites

- www.mathworks.com/hardware-support/simulink-desktop-real-time.html

Install Real-Time Kernel

The Simulink Desktop Real-Time software requires a real-time kernel that interfaces with the operating system. The Simulink Desktop Real-Time kernel assigns the highest priority of execution to your real-time executable, which allows it to run without interference at the selected sample rate. During real-time execution, the kernel intervenes to give the model priority to use the CPU to execute each model update at the prescribed sample times. Once a model update completes, the kernel releases the CPU to run other operating system-based applications that need servicing.

Install the Kernel Using MATLAB

You must install the kernel before you can run a Simulink Desktop Real-Time application. During software installation, the Simulink Desktop Real-Time software is copied onto your hard drive, but the Simulink Desktop Real-Time kernel is not automatically installed into the operating system.

Installing the kernel configures it to start running in the background each time you start your computer. The following procedure describes how to use the command `sldrtkernel -install`. (You can also use the command `sldrtkernel -setup` instead.) To install the kernel:

- 1 In the MATLAB Command Window, type:

```
sldrtkernel -install
```

The MATLAB Command Window displays one of these messages:

```
You are going to install the Simulink Desktop Real-Time kernel.  
Do you want to proceed? [y] :
```

or:

```
There is a different version of the Simulink Desktop Real-Time kernel installed.  
Do you want to update to the current version? [y] :
```

- 2 Type **y** to continue installing the kernel, or **n** to cancel installation without changing the installation.

If you type **y**, the MATLAB environment installs the kernel and displays the message:

```
The Simulink Desktop Real-Time kernel has been successfully installed.
```

- 3 If a message appears asking you to restart your computer, do so before attempting to use the kernel, or your Simulink Desktop Real-Time model does not run.
- 4 After installing the kernel, check the installation by typing:

```
rtwho
```

The MATLAB Command Window displays a message that shows the kernel version number, followed by timer, driver, and other information.

Once the kernel is installed, it remains idle. You can leave it installed. While the kernel is idle, the operating system controls the execution of standard applications, such as Internet browsers, word processors, and the MATLAB environment. The kernel becomes active when you begin execution of your model, and becomes idle again after model execution completes.

Uninstall the Kernel

If you encounter problems with Simulink Desktop Real-Time software, you can uninstall the kernel. Once uninstalled, the kernel is no longer active. The kernel executable file remains on your hard drive so that you can later reinstall it.

Uninstall the Kernel Using MATLAB

To uninstall the kernel from MATLAB:

- 1 In the MATLAB Command Window, type:

```
sldrkernel -uninstall
```

The MATLAB Command Window displays the message:

```
You are going to uninstall the Simulink Desktop Real-Time kernel.  
Do you want to proceed? [y]:
```

- 2 Type `y` to continue uninstalling the kernel, or `n` to stop uninstalling without changing the installation.

If you type `y`, the MATLAB environment uninstalls the kernel by removing it from memory, then displays the message:

```
The Simulink Desktop Real-Time kernel has been successfully uninstalled.
```

- 3 After uninstalling the kernel, check that it was uninstalled. Type:

```
rtwho
```

The MATLAB Command Window displays the message:

```
Simulink Desktop Real-Time installation is not complete.  
Please type 'sldrkernel -setup' to complete the installation.  
Type 'help sldrkernel' for more information.
```

Uninstall the Kernel Using Development Computer Command Line

Uninstalling the MATLAB environment does not uninstall the Simulink Desktop Real-Time kernel. If you uninstalled the MATLAB environment without uninstalling the kernel, open a development computer command window and type:

```
sldrkernel -uninstall
```

The `sldrkernel` program uninstalls the kernel by removing it from memory, then displays the message:

```
The Simulink Desktop Real-Time kernel uninstalled successfully.
```

This procedure works only with the Windows operating system.

See Also

`sldrkernel`

Run Confidence Test

Simulink Desktop Real-Time includes several example models you can use to test your installation. These models are preconfigured with settings such as target and scope settings, sample time, and integration algorithm. To see these models, type `sldrtexamples` in the MATLAB Command Window.

Note

- You can run examples in **Run in Kernel** mode (initial setting) or **Connected IO** mode.
 - You cannot run a Simulink Desktop Real-Time model in rapid accelerator mode.
-

Once you have finished installing the Simulink Desktop Real-Time software and kernel, test the installation by running the model `sldrtex_vdp`. If you change your installation, repeat this test to confirm that the Simulink Desktop Real-Time software is still working. To open the example model, type `sldrtex_vdp` in the MATLAB Command Window, or start MATLAB Help, open **Simulink Desktop Real-Time**, and choose **Examples > Real-Time Van der Pol Simulation**.

Run the Model `sldrtex_vdp`

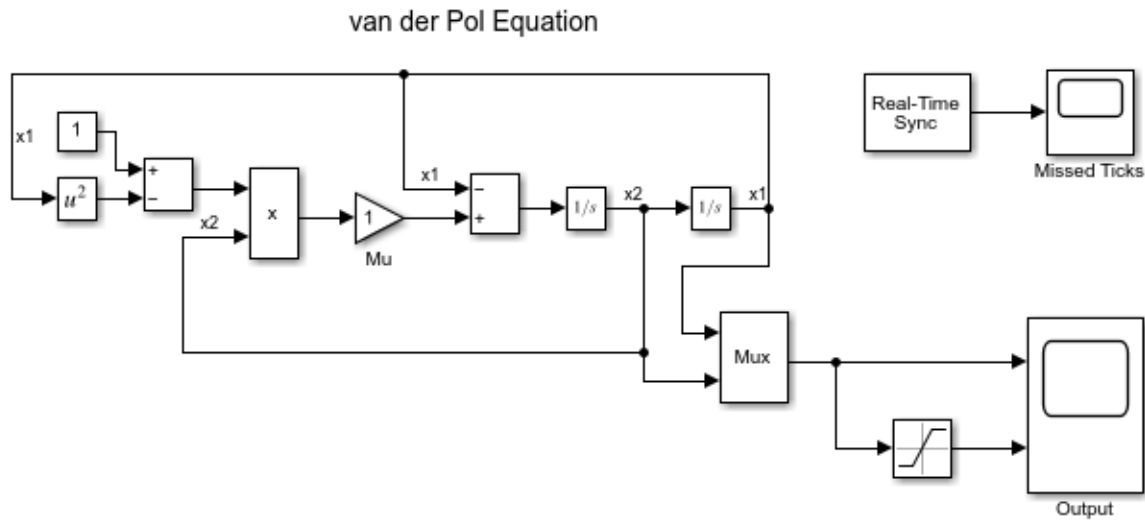
The model `sldrtex_vdp` does not have I/O blocks. This design lets you can run this model regardless of the I/O boards in your computer. Running this model tests the installation by running Simulink Coder code generation software, Simulink Desktop Real-Time software, and the Simulink Desktop Real-Time kernel.

After you have installed the Simulink Desktop Real-Time kernel, you can test the entire installation by building and running a real-time application. The Simulink Desktop Real-Time software includes the model `sldrtex_vdp`, which already has Simulink Coder options selected for you.

- 1 In the MATLAB Command Window, type:

```
sldrtex_vdp
```

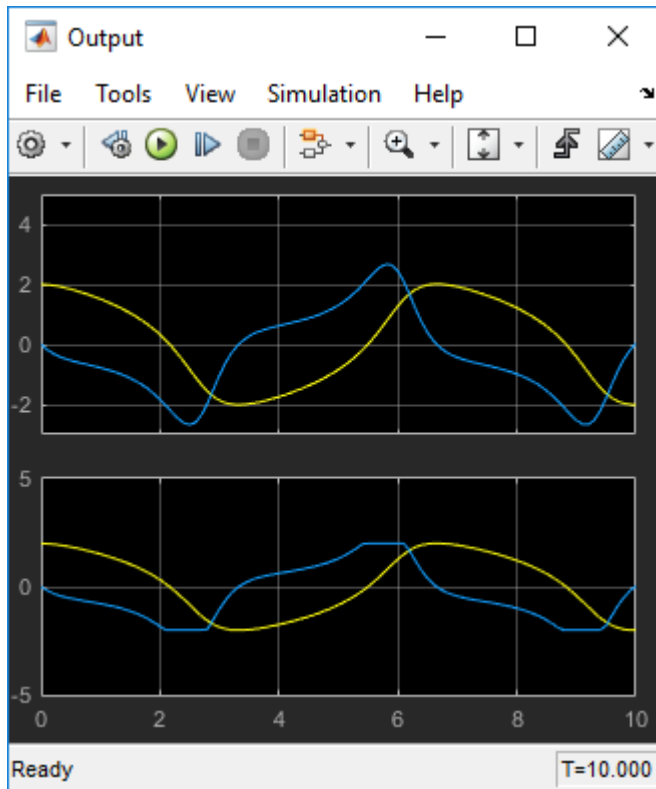
The Simulink Editor opens and displays the model `sldrtex_vdp`.



To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.

- 2 To build the model, connect to the Simulink Desktop kernel, change to **Run in Kernel** mode simulation, and run the model in real time, on the **Desktop Real-Time** tab, click **Run in Real Time**.

After building the model and displaying messages in the Diagnostic Viewer, the target application begins running.



Tip You can execute the build, connect, and run operations as individual steps. On the **Desktop Real-Time** tab, from the **Run in Real Time** button, select available **Step by Step Commands**.

- 3 To stop the simulation before it ends, on the **Desktop Real Time** tab, click **Stop**.

The real-time application stops running. The Scope window stops displaying the output signals.

Display Status Information

The Simulink Desktop Real-Time software provides the command `rtwho` for displaying the kernel version number, followed by timer, driver, and other information. To see this information, in the MATLAB Command Window type:

```
rtwho
```

The command displays several lines of information in the MATLAB Command Window. Some possible lines and their interpretations are:

```
TIMERS:  Number   Period  Running
          1       0.01    Yes
```

The indicated timers exist on your system with the period and run status shown for each timer.

```
DRIVERS:           Name      Address  Parameters
                Humusoft AD512  0x300   []
                ecg      0       []
```

The indicated device drivers are installed on your system at the address and with the parameters shown for each driver.

Examples Library

The examples library includes models with preset values and dialog boxes. These models include simple signal processing and simple control examples that use no I/O blocks, use A/D blocks only, and use both A/D and D/A blocks.

To run an example that uses I/O blocks, you must configure the block to match the I/O board installed in your computer. There are some simulation mode limitations.

- You can run examples in **Run in Kernel** mode (initial setting) or **Connected IO** mode.
- You cannot run a Simulink Desktop Real-Time model in rapid accelerator mode.

To see these models from the MATLAB environment:

- 1 Type `sldrtexamples` in the MATLAB Command Window.
- 2 In the Simulink Desktop Real-Time Examples window, from the list, select the example to open it.

See Also

More About

- “Troubleshoot Vendor Software Missing Issues” on page 6-7
- “Troubleshoot sldrttext Incorrect Version Error” on page 6-3
- “Troubleshoot Delayed or Missing Scope Output” on page 6-4
- “Troubleshoot Signals Not Plotted in Scope Blocks” on page 6-5

Basic Procedures

- “Prepare for Real-Time Execution” on page 3-2
- “Create a Real-Time Application” on page 3-4
- “Create a Simulink Model” on page 3-5
- “File System I/O” on page 3-9
- “Configure a Model for Simulink Desktop Real-Time” on page 3-10
- “Simulate Model in Connected IO Mode” on page 3-15
- “Set Run in Kernel Mode Code Generation Parameters” on page 3-17
- “Prepare Run in Kernel Mode Application” on page 3-20
- “Set Run in Kernel Mode (External Mode) Scope Parameters” on page 3-21
- “Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time” on page 3-24
- “Execute Real-Time Application in Run in Kernel Mode by Using Step by Step Commands” on page 3-26
- “Execute Real-Time Application with S-Functions in Run in Kernel Mode” on page 3-28
- “Run Application from MATLAB Command Line” on page 3-30
- “Inspect Simulink® Desktop Real-Time™ Signals with Simulation Data Inspector” on page 3-32
- “Signal Logging to the Workspace” on page 3-35
- “Set Scope Parameters for Logging to Workspace” on page 3-36
- “Set Run in Kernel Mode Properties for Logging to Workspace” on page 3-38
- “Plot Signal Data Logged to Workspace” on page 3-40
- “Signal Logging to a File” on page 3-42
- “Set Scope Parameters for Logging to File” on page 3-43
- “Set Run in Kernel Mode Properties for Logging to File” on page 3-45
- “Set Run in Kernel Mode Data Archiving Parameters” on page 3-47
- “Plot Signal Data Logged to File” on page 3-49
- “Tunable Block Parameters and Tunable Global Parameters” on page 3-51
- “Tune Block Parameters by Using the Block Dialog Box” on page 3-54
- “Tune Block Parameters with Data Navigation” on page 3-57
- “Sweep MATLAB Variables with MATLAB Scripting” on page 3-62

Prepare for Real-Time Execution

In this section...

“Prepare I/O Devices” on page 3-2

“Prepare Real-Time Application” on page 3-2

To observe how Simulink models respond to real-world behavior, connect the real-time application to physical I/O devices. You have access to a library of I/O driver blocks that provide connections between devices and applications. To prepare for real-time execution, install I/O devices in your computer and select the corresponding Simulink Desktop Real-Time library blocks.

Actions to consider when preparing for real-time execution include:

- 1 “Use I/O Boards” on page 4-2
- 2 “Use I/O Driver Blocks” on page 4-5
- 3 “Use Analog I/O Drivers” on page 4-9
- 4 “Use Vector CAN Drivers” on page 4-12
- 5 “Custom I/O Driver Basics” on page A-2
- 6 “Simulink Desktop Real-Time Pane”
- 7 “Execution Time Measurement and Block Profiling” on page 5-32

Prepare I/O Devices

- 1 Choose I/O plugin modules from www.mathworks.com/hardware-support/simulink-desktop-real-time.html.
- 2 Acquire the modules and install them in your computer.
- 3 Refer to the vendor documentation for vendor-specific requirements.

If the hardware requires the installation of vendor software, install the vendor software on your computer.

- 4 Restart your computer and, from the MATLAB Command Window, start the Simulink Desktop Real-Time kernel.

Prepare Real-Time Application

- 1 Replace Simulink I/O blocks with Simulink Desktop Real-Time blocks that represent the functionality of your I/O modules.
- 2 Open the dialog box for each block and associate the block with the driver for the corresponding I/O module.

Set the other block parameters as required by your model.

- 3 To configure the model for Simulink Desktop Real-Time code generation, in the Simulink Editor, from the **Apps** tab, select **Desktop Real-Time**.

This operation selects the `sldrt.tlc` code generation target and sets other configuration parameters for compatibility with Simulink Desktop Real-Time.

- 4 Set the Simulink Desktop Real-Time configuration parameters as required by your model.

- 5 If the performance of your model in **Run in Kernel** mode does not meet your system requirements, add Execution Time and Timestamp blocks for analysis. See “Execution Time Measurement and Block Profiling” on page 5-32.

The next step is to set the simulation mode to **Connected IO** mode or **Run in Kernel** mode to attain the required sample rate, and then run the simulation.

See Also

Execution Time | Timestamp

More About

- “Create a Real-Time Application” on page 3-4
- “Execution Time Measurement and Block Profiling” on page 5-32

External Websites

- www.mathworks.com/hardware-support/simulink-desktop-real-time.html

Create a Real-Time Application

A Simulink model is a graphical representation of your physical system. You create a Simulink model for a non-real-time simulation of your system, and then you use the Simulink model to create a real-time application. This example uses `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...  
    'sldrt','examples','sldrtex_model')))
```

Note You cannot run a Simulink Desktop Real-Time model in rapid accelerator mode.

To run `sldrtex_model` in **Connected IO** mode:

- 1 “Configure a Model for Simulink Desktop Real-Time” on page 3-10
- 2 “Simulate Model in Connected IO Mode” on page 3-15

To run `sldrtex_model` as a real-time application:

- 1 “Set Run in Kernel Mode Code Generation Parameters” on page 3-17
- 2 “Prepare Run in Kernel Mode Application” on page 3-20
- 3 “Set Run in Kernel Mode (External Mode) Scope Parameters” on page 3-21
- 4 “Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time” on page 3-24

See Also

More About

- “Create a Simulink Model” on page 3-5
- “Run Application from MATLAB Command Line” on page 3-30

Create a Simulink Model

Create a simple model of a damped square-wave generator. You can use this model as an example to learn other capabilities that are useful with Simulink Desktop Real-Time software.

- 1 In the MATLAB **Home** tab, click the **Simulink** button.
- 2 Click **Blank Model**, and then **Create Model**.

An empty Simulink Editor opens.

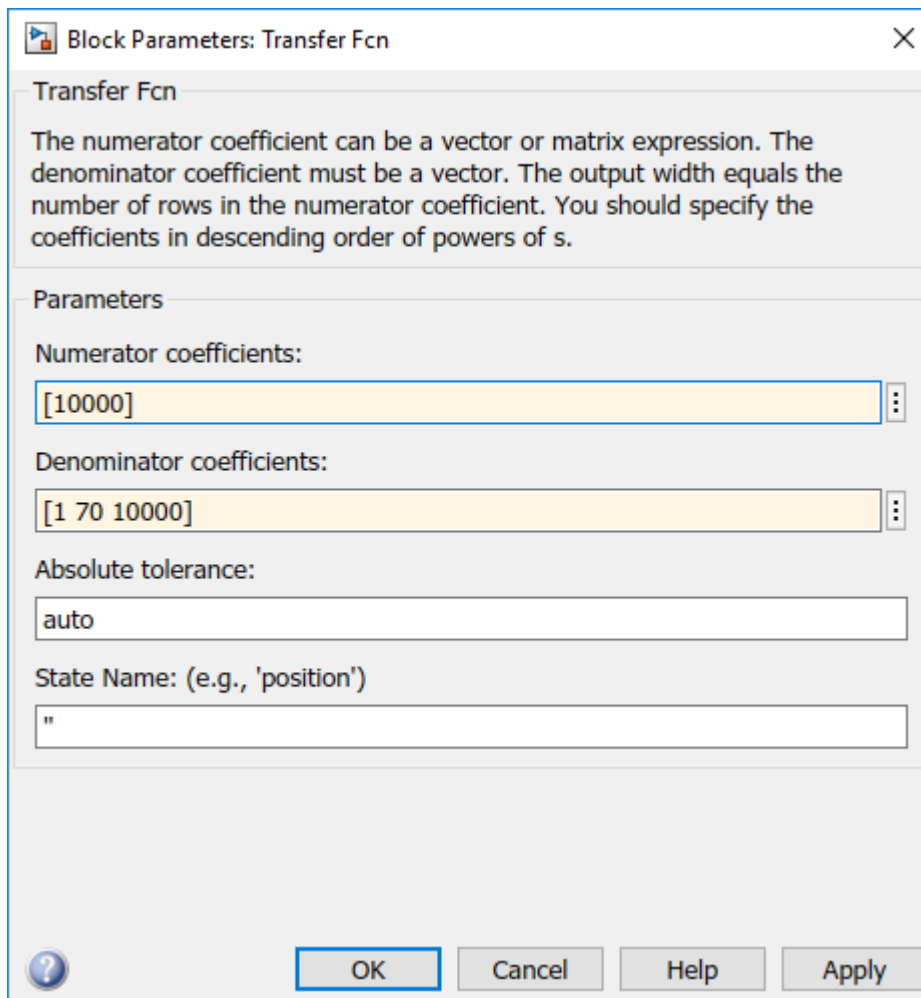
- 3 On the **Simulation** tab, click **Library Browser**.
- 4 In the Library Browser:
 - Select **Simulink > Sources**, and then add a Signal Generator block to the model.
 - Select **Simulink > Continuous**, and then add a Transfer Fcn block to the model.
 - Select **Simulink > Sinks**, and then add a Scope block to the model.
- 5 Make the following block-to-block connections:
 - Signal Generator output to Transfer Fcn input
 - Transfer Fcn output to Scope input
- 6 Double-click the Transfer Fcn block. The Block Parameters dialog box opens. In the **Numerator** text box, enter:

```
[10000]
```

In the **Denominator** text box, enter:

```
[1 70 10000]
```

Click **OK**.



- 7 Double-click the Signal Generator block. From the **Wave form** list, select square.

In the **Amplitude** text box, enter:

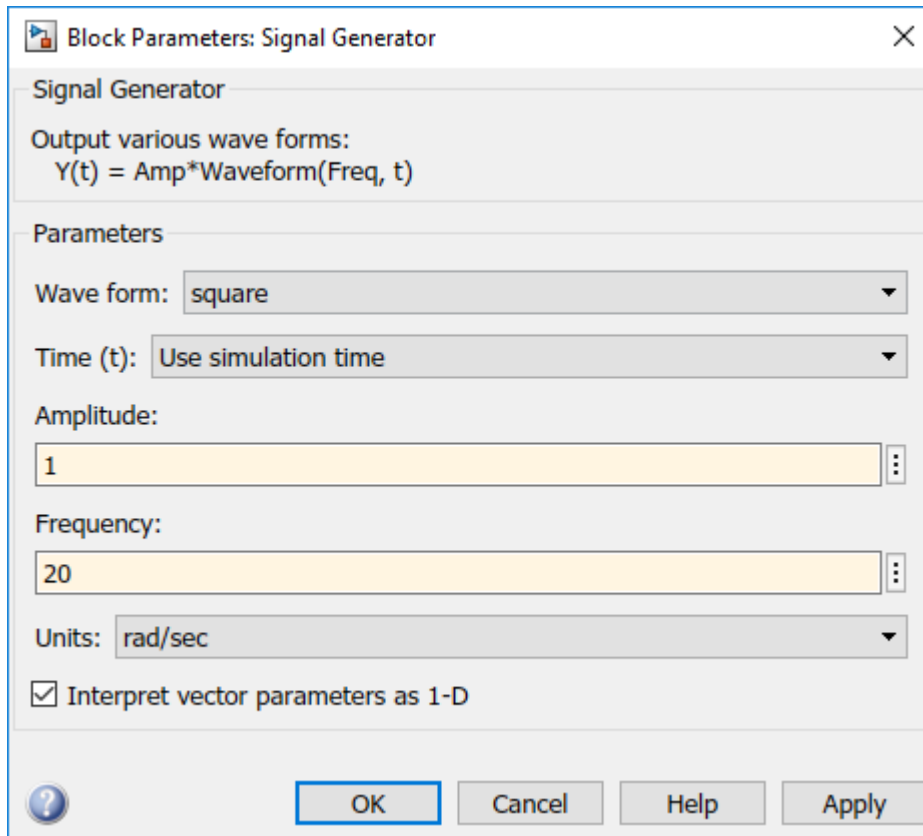
1

In the **Frequency** text box, enter:

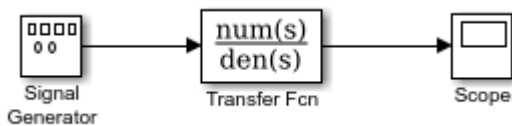
20

From the **Units** list, select rad/sec.

Click **OK**.



The completed Simulink block diagram looks like the figure.



- 8 In the Simulink Editor, on the **Simulation** tab, click **Save > Save as**. In the **File name** text box, enter a file name for your Simulink model and click **Save**. For example, type:

```
sldrtex_model
```

The Simulink software saves your model in the file `sldrtex_model`.

The Simulink Desktop Real-Time software supports model referencing. See “Model Reference Basics”.

The Simulink Desktop Real-Time software supports file I/O, with constraints. See “File System I/O” on page 3-9.

To specify a default Simulink Desktop Real-Time configuration set for your model, see “Specify a Default Configuration Set” on page 3-10. If you activate this configuration set for your model, you can build your real-time application later without setting additional configuration parameters.

To configure your model manually, see “Enter Configuration Parameters Manually” on page 3-11.

See Also

More About

- “Model Reference Basics”
- “File System I/O” on page 3-9
- “Specify a Default Configuration Set” on page 3-10
- “Enter Configuration Parameters Manually” on page 3-11

File System I/O

You can use file I/O blocks in **Connected IO** mode or accelerator mode simulation in real time. In these modes, the real-time kernel does not perform I/O, Simulink itself does.

When run in **Run in Kernel** mode, the Simulink Desktop Real-Time software does not include a file system. Therefore, in a Simulink Desktop Real-Time model, you cannot use blocks that generate file I/O calls such as `fopen` or `fprintf`. Examples of such blocks are To File and From File. To access files in **Run in Kernel** mode, use the Packet Input, Packet Output, Stream Input, or Stream Output blocks, and select the driver **Standard Devices > File**.

If a Simulink Desktop Real-Time model contains an I/O block, an error can occur when you try to compile or use **Run in Kernel** mode with the model. Even if no error occurs, the block has no effect on either simulation or code execution.

To log signal data without a file system, use the techniques described in “Signal Logging to the Workspace” on page 3-35. For information about using **Run in Kernel** mode to execute a Simulink Desktop Real-Time application, see “Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time” on page 3-24.

See Also

More About

- “Signal Logging to the Workspace” on page 3-35
- “Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time” on page 3-24

Configure a Model for Simulink Desktop Real-Time

After you create a Simulink model, you can enter configuration parameters for the model. These parameters control many properties of the model for simulation and code generation.

A configuration set is a named set of values for model parameters, such as solver type and simulation start or stop time. Every Simulink model is created with a default configuration set, called **Configuration**, that initially specifies default values for the model parameters. You can then create additional configuration sets and associate them with the model. For more information about Simulink configuration, see “Manage Configuration Sets for a Model”.

The easiest way to specify configuration parameters for a Simulink Desktop Real-Time model is to assign the default Simulink Desktop Real-Time configuration set programmatically, as described in “Specify a Default Configuration Set” on page 3-10. You can also set parameters manually, as described in “Enter Configuration Parameters Manually” on page 3-11.

Specify a Default Configuration Set

After you create a Simulink model, you can use the `sldrtconfigset` function to specify a default Simulink Desktop Real-Time configuration set for the model. Usually, using `sldrtconfigset` provides the configuration parameter values that the model requires.

The following procedure uses the model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot, 'toolbox', ...
    'sldrt', 'examples', 'sldrtex_model')))
```

It assumes that you have already loaded that model (see “Create a Simulink Model” on page 3-5).

- 1 If you have not already saved the model, on the **Simulation** tab, click **Save > Save as**. In the **File name** text box, enter a file name for your Simulink model and click **Save**. For example, type:

```
sldrtex_model
```

The Simulink software saves your model in the file `sldrtex_model`.

- 2 In the MATLAB Command Window, type:

```
sldrtconfigset('sldrtex_model')
```

The default Simulink Desktop Real-Time configuration set, `SimulinkDesktopRealTime`, is now active for the `sldrtex_model` model.

- 3 Save the model.

For a description of how to build your Simulink Desktop Real-Time application, see “Create a Real-Time Application” on page 3-4.

To revert to the default configuration set, `Configuration`, or other configuration set you have for the model, use Model Explorer. For a description of how to use Model Explorer, see the Simulink documentation.

Your model uses a Simulink Desktop Real-Time configuration set when you change the **System target file** value to a Simulink Desktop Real-Time one, such as `sldrt.tlc` or `sldrtert.tlc`. The software creates the Simulink Desktop Real-Time configuration set only if one does not exist.

Enter Configuration Parameters Manually

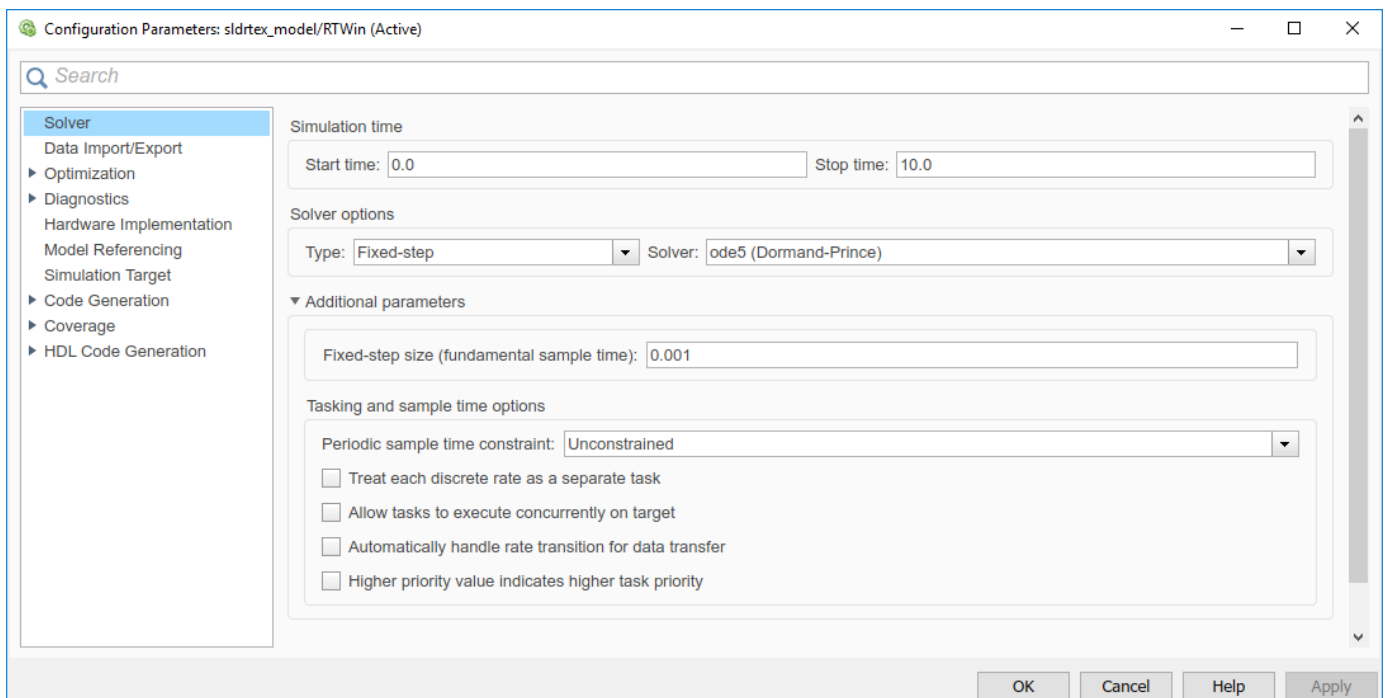
The configuration parameters give information to Simulink software for running a simulation.

This procedure uses the model `sldrtext_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...
    'sldrt','examples','sldrtext_model')))
```

It assumes that you have already loaded that model.

- 1 On the **Desktop Real-Time** tab, click **Hardware Settings**. In the Configuration Parameters dialog box, click the **Solver** tab.
- 2 In the **Start time** field, enter `0.0`. In the **Stop time** field, enter the amount of time you want your model to run. For example, enter `10.0` seconds.
- 3 From the **Type** list, select **Fixed-step**. Simulink Coder does not support variable step solvers.
- 4 From the **Solver** list, select a solver. For example, select the general-purpose solver `ode5` (Dormand-Prince).
- 5 Under **Additional options**, in the **Fixed step size** field, enter a sample time. For example, enter `0.001` seconds for a sample rate of 1000 samples/second.
- 6 Leave the parameter **Treat each discrete rate as a separate task** cleared. (For models with blocks that have different sample times, select this parameter.)




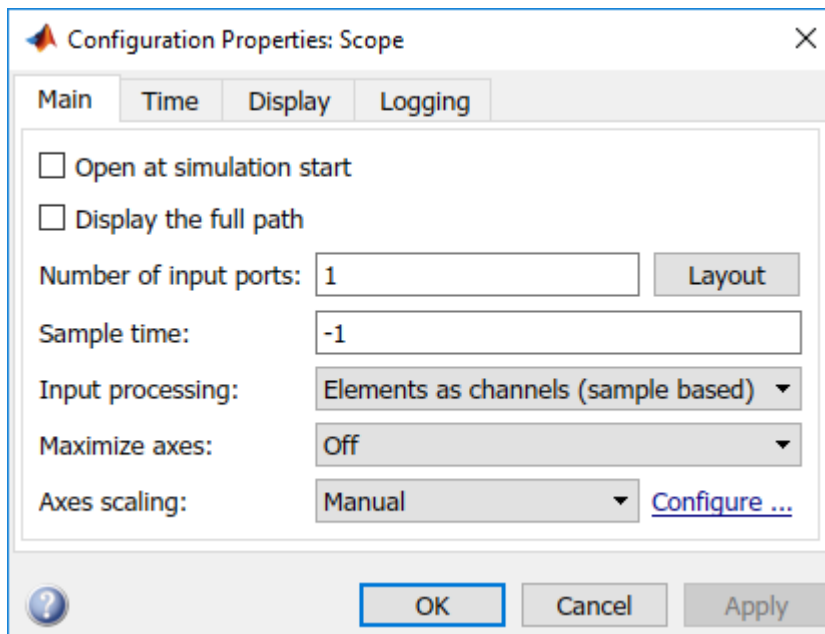
- 7 Click **OK**.

Enter Scope Parameters for Signal Tracing

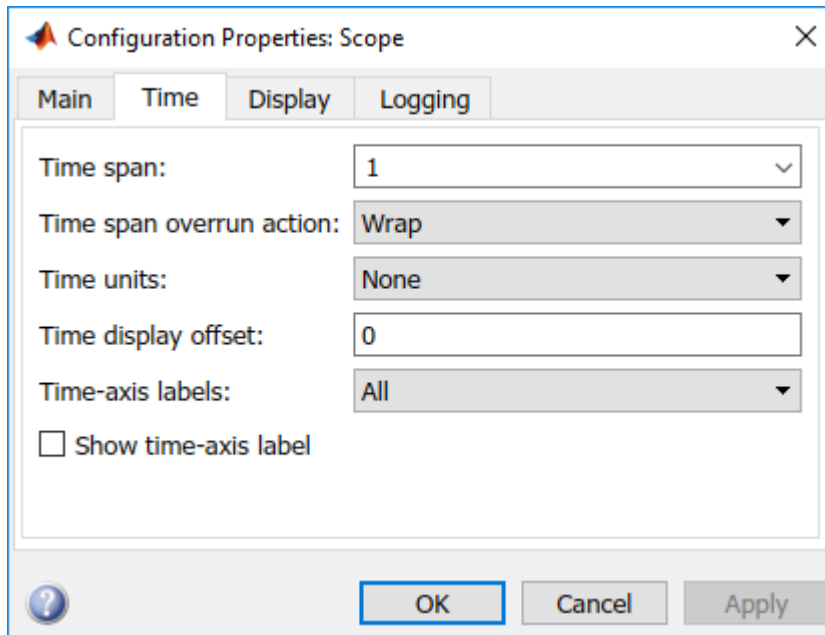
You enter or change scope parameters to specify the x-axis and y-axis in a Scope window. Other properties include the number of graphs in one Scope window and the sample time for models with discrete blocks.

After you add a Scope block to your Simulink model, you can enter the scope parameters for signal tracing:

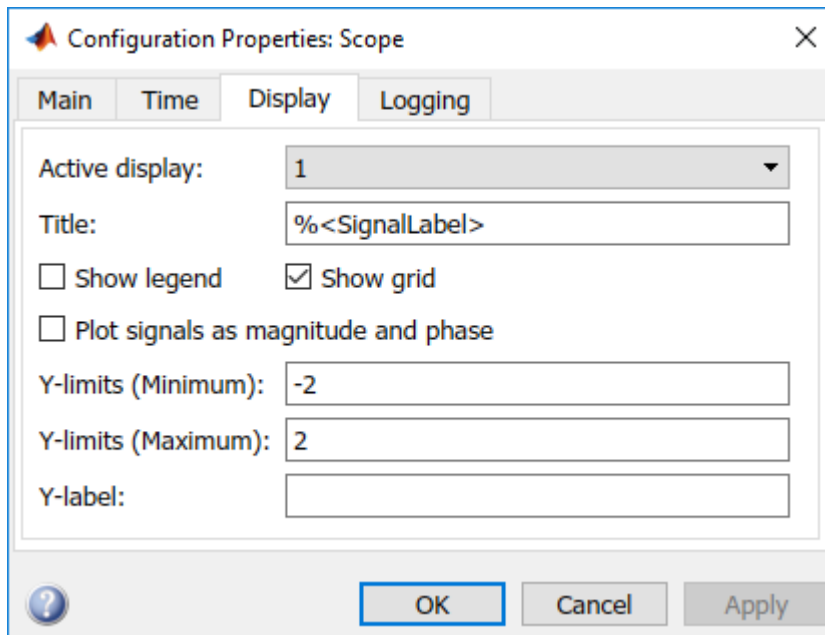
- 1 In the Simulink Editor, double-click the Scope block.
- 2 On the toolbar, click the **Parameters** button .
- 3 Click the **Main** tab. In the **Sample time** text box, enter -1, which indicates that this block inherits its value from its parent model. If you have discrete blocks in your model, enter the **Fixed step size** value that you entered in the Configuration Parameters dialog box.



- 4 Click the **Time** tab. In the **Time span** box, enter 1.



- 5 Click the **Display** tab. In the **Y-min** and **Y-max** text boxes, enter the range for the y-axis in the Scope window. For example, enter -2 and 2.



- 6 Click **OK**.

See Also

More About

- “Manage Configuration Sets for a Model”

- “Create a Real-Time Application” on page 3-4
- “Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time” on page 3-24
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Simulate Model in Connected IO Mode

To run a real-time simulation of a Simulink Desktop Real-Time model, you can use Simulink **Connected IO** mode. This procedure uses the model `sldrtext_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...
    'sldrt','examples','sldrtext_model')))
```

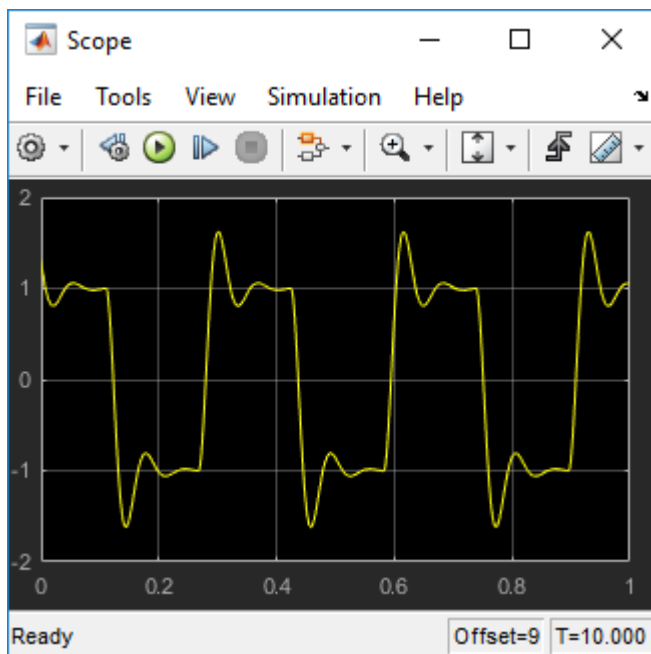
Load the model before you start the procedure.

- 1 In the Simulink Editor, double-click the Scope block.

The Simulink software opens a Scope window with an empty graph.

- 2 On the **Desktop Real-Time** tab, select **Mode > Connected IO** for the simulation mode.
- 3 Open the block parameters of the Real-Time Sync block in your Simulink Desktop Real-Time model.
- 4 To prevent missed ticks, set values for the **Sample Time** and **Maximum Missed Ticks** block parameters.
- 5 To begin simulation, on the **Desktop Real-Time** tab, click **Run in Real-Time**.

The Simulink software runs the simulation and plots the signal data in the Scope window.



- 6 To stop the simulation before it ends, on the **Simulation** tab, click **Stop**.

The real-time application stops running. The Scope window stops displaying the output signals.

See Also

More About

- “Real-Time Execution in Run in Kernel Mode” on page 1-5
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Set Run in Kernel Mode Code Generation Parameters

After you create a Simulink model, you can enter simulation parameters. Simulink Coder uses these parameters for creating C code and building a real-time application.

This procedure uses the model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...  
    'sldrt','examples','sldrtex_model')))
```

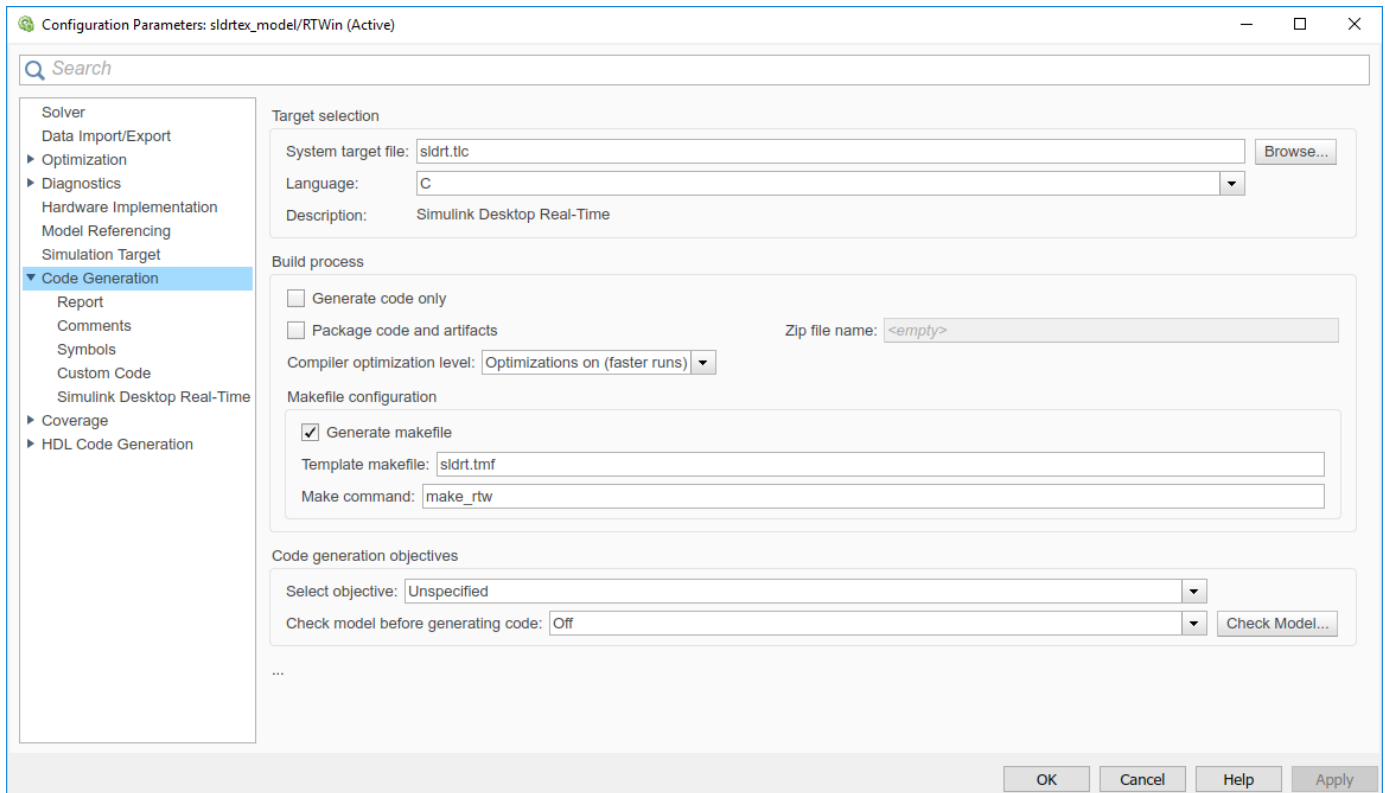
It assumes that you have already loaded that model.

- 1 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Hardware Settings**.
- 2 In the Configuration Parameters dialog box, click the **Code Generation** node.
- 3 In the **Target selection** section, click the **Browse** button at the **System target file** list.
- 4 In the **System target file** browser, select the system target file for building a Simulink Desktop Real-Time application, `sldrt.tlc`, and click **OK**.

The dialog box enters the system target file `sldrt.tlc`, the template makefile `sldrt.tmf`, and the make command `make_rtw` into the **Code Generation** pane.

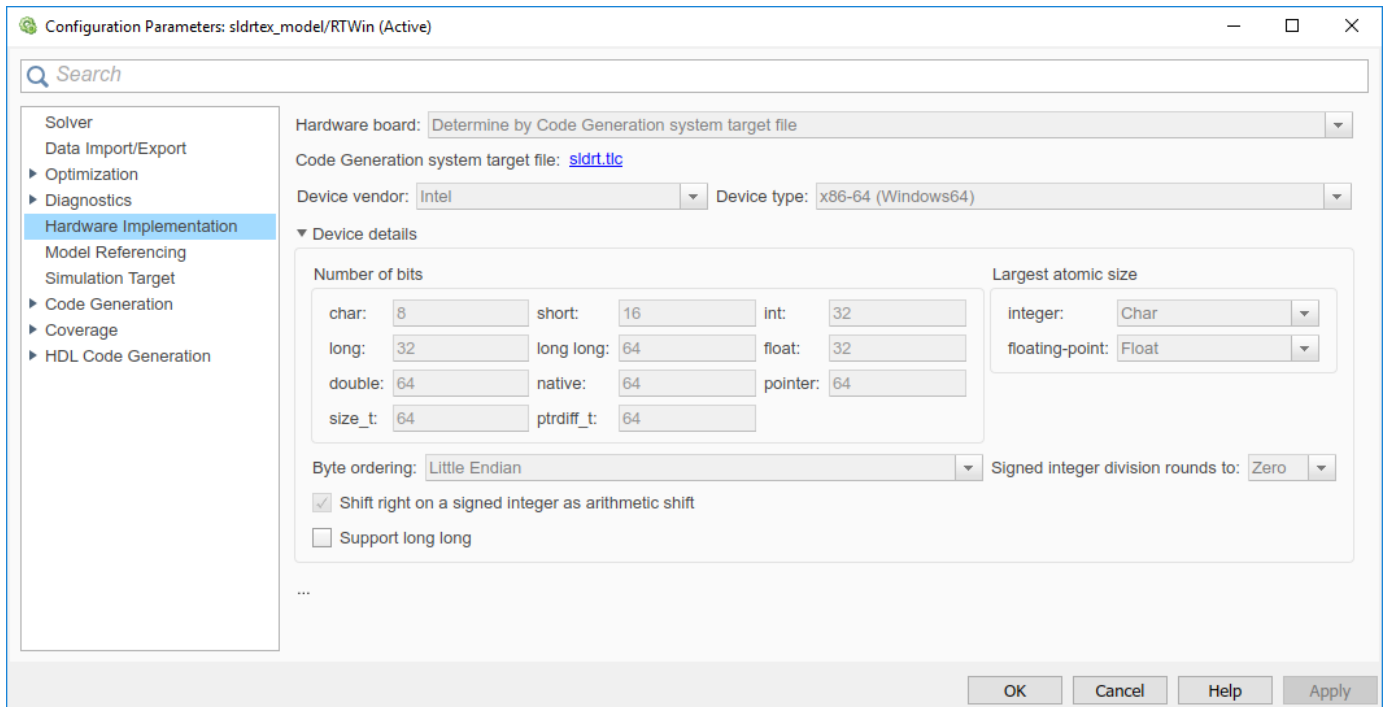
If you have the Embedded Coder® product, you can build an ERT target application. To build an ERT target application, in the **Target selection** section, click the **Browse** button at the **System target file** list. Click `sldrtert.tlc`, and then click **OK**.

Although not visible in the **Code Generation** pane, when you click **OK** you also configure the external target interface MEX file `sldrtext`. This file allows **Run in Kernel** mode to pass new parameters to the real-time application and to return signal data from the real-time application. The data is displayed in Scope blocks or saved with signal logging.



Do not set **Default parameter behavior** to **Inlined** on the **Optimization** node under **Code Generation**. Inlining parameters is for custom targets when you want to reduce the amount of RAM or ROM with embedded systems. Also, if you select inlining parameters, you disable the parameter tuning feature. Do not inline parameters because PCs have more memory than embedded systems.

- 5 Click the **Hardware Implementation** node. The default values are derived from the architecture of the development computer. For example, for a 64-bit Intel® machine, they are:
 - **Device vendor** — Intel
 - **Device type** — x86-64



6 Click **OK**.

See Also

More About

- “Real-Time Execution in Run in Kernel Mode” on page 1-5
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Prepare Run in Kernel Mode Application

In **Run in Kernel** mode, you must first create an executable target application. The Simulink Coder code generation software creates C code from your Simulink model. The bundled C compiler compiles and links that C code into a real-time application.

This procedure uses the model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...  
    'sldrt','examples','sldrtex_model')))
```

It assumes that you have already loaded that model.

- 1 In the Simulink Editor, from the **Apps** tab, click **Desktop Real-Time**.

This operation selects configuration parameters for use by the Simulink Coder code generation software. See “Set Run in Kernel Mode Code Generation Parameters” on page 3-17.

- 2 On the **Desktop Real-Time** tab, click **Run in Real Time**.

- The Simulink Coder code generation software creates the C code source files `sldrtex_model.c` and `sldrtex_model.h`.
- The build process creates the makefile `sldrtex_model.mk` from the template makefile `sldrt.tmf`.
- The build process creates the real-time application by using `sldrtex_model.mk`. On Windows, the build process creates the binary file `sldrtex_model.rwx64`. On Mac OS, it creates the binary file `sldrtex_model.rxm64`.

The binary file `sldrtex_model.rx*64` is referred to as a real-time application. You can run the real-time application with the Simulink Desktop Real-Time kernel.

After you create a real-time application, you can exit MATLAB, and restart MATLAB again, and then connect and run the executable without rebuilding your code. For more information, see “Execute Real-Time Application in Run in Kernel Mode by Using Step by Step Commands” on page 3-26.

See Also

More About

- “Real-Time Execution in Run in Kernel Mode” on page 1-5
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Set Run in Kernel Mode (External Mode) Scope Parameters

Connected IO mode and accelerator modes run the simulation algorithm in Simulink and access the external hardware by using drivers running in operating system kernel mode. The Simulink block diagram is a user interface to your real-time application.

Run in Kernel mode connects your Simulink model to your real-time application. You can use the Simulink block diagram as a user interface as you can in **Connected IO** mode or accelerator mode.

This procedure uses the model `sldrtext_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot, 'toolbox', ...
    'sldrt', 'examples', 'sldrtext_model')))
```

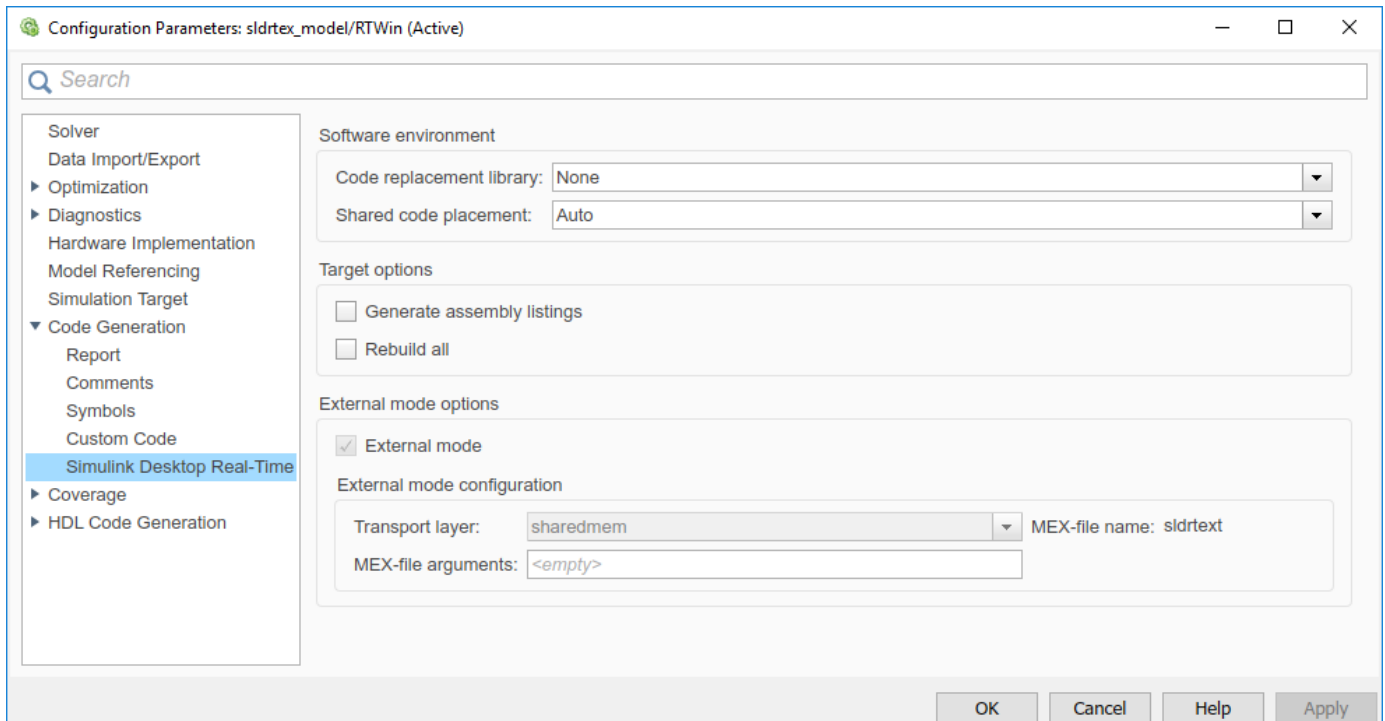
It assumes that you have already loaded that model.

After you have created a real-time application, you can enter scope parameters for signal tracing with Simulink **Run in Kernel** mode:

- 1 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Prepare > Hardware Settings**.
- 2 In the Configuration Parameters dialog box, select the **Code Generation > Simulink Desktop Real-Time** node.
- 3 If you select the **External mode** check box, your changes affect the real-time application.

Check that the **MEX-file name** label has an entry of `sldrtext`. The MEX-file `sldrtext.mex*` is supplied with the Simulink Desktop Real-Time software. This file works with Simulink **Run in Kernel** mode and supports uploading signal data and downloading parameter values.

Click **OK**.



- 4 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Prepare > Control Panel**. In the External Mode Control Panel, click the **Signal & Triggering** button.
- 5 Select the **Select all** check box. From the **Source** list, select **manual**. From the **Mode** list, select **normal**.

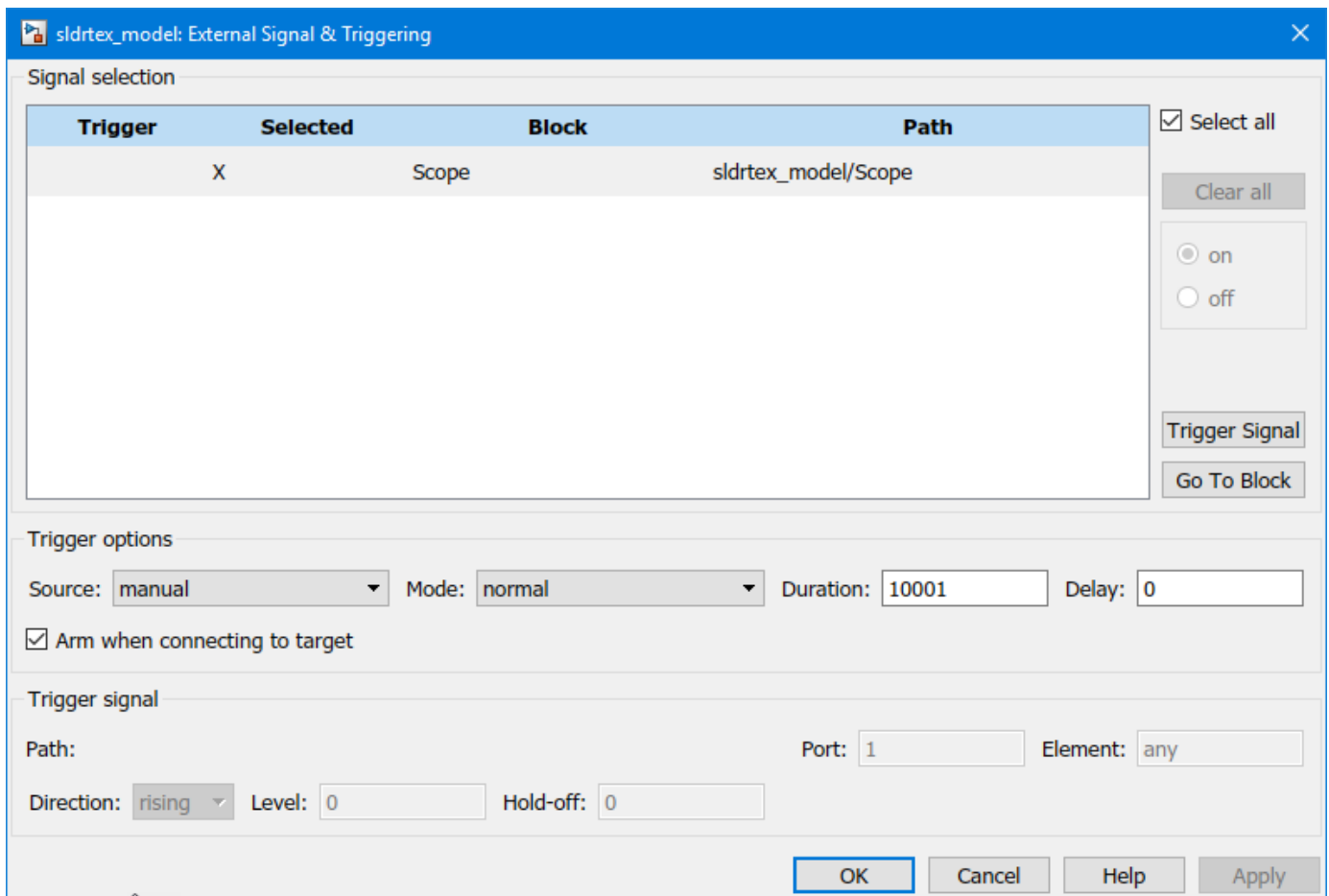
The X under **Signal selection** indicates that a signal is tagged for data collection. T indicates that the signal is tagged as a trigger signal.

- 6 In the **Duration** field, enter the number of sample points in a data buffer. For example, to specify a sample rate of 1000 samples/second and a stop time of 10 seconds, enter:

10000

- 7 Select the **Arm when connecting to target** check box.

If you do not select this check box, data is not displayed in the Scope window.



- 8 Click **Close**.

See Also

More About

- “Real-Time Execution in Run in Kernel Mode” on page 1-5
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time

After you build the real-time application, you can run your model in real time. In **Run in Kernel** mode, you execute your real-time application to observe the behavior of your model in real time with the generated code.

Note You cannot run a real-time application in **Rapid Accelerator** mode.

This procedure uses the model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...  
    'sldrt','examples','sldrtex_model')))
```

It shows how to use **Run in Real Time** to complete these operations with one click:

- **Build** — Create a real-time application from the model.
- **Connect** — Establish a connection between the model and the kernel. This connection permits exchange of commands, parameters, and logged data.
- **Start** — Start execution of the application in real time.

- 1 Open the model `sldrtex_model`.
- 2 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Run in Real Time**.

Simulink builds the real-time application, changes to external mode simulation, connects the model and kernel, and runs the simulation. You can accomplish these and other simulation operations as individual steps. For more information, see “Execute Real-Time Application in Run in Kernel Mode by Using Step by Step Commands” on page 3-26.

- 3 To stop the simulation before it ends, on the **Desktop Real-Time** tab, click the **Stop**.

In this example, the Scope window displays 1000 samples in 1 second, increases the time offset, and then displays the samples for the next 1 second.

Transfer of data is less critical than calculating the signal outputs at the selected sample interval. Therefore, data transfer runs at a lower priority in the CPU time that remains after real-time application computations are performed. As a result, data points can be omitted from the Scope block display.



See Also

More About

- "Execute Real-Time Application in Run in Kernel Mode by Using Step by Step Commands" on page 3-26
- "Simulate Model in Connected IO Mode" on page 3-15
- "Troubleshoot Missing Desktop Real-Time Tab" on page 6-2

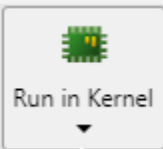
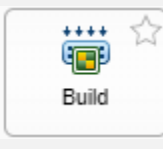


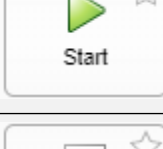

Execute Real-Time Application in Run in Kernel Mode by Using Step by Step Commands

After you build the real-time application, you can run your model in real time. In **Run in Kernel** mode, you execute your real-time application to observe the behavior of your model in real time by using the generated code.

This procedure uses the model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...
    'sldrt','examples','sldrtex_model')))
```

It shows how to use Step by Step Commands to complete simulation operations as individual steps.

Buttons	Step by Step Command Operations
 Run in Kernel	Select Run in Kernel mode for simulation.
 Build	Build the real-time application.
 Connect	Connects the model and kernel.
 Disconnect	Disconnect the real-time application from the model and kernel.
 Start	Start running the real-time simulation.
 Stop	Stop running the real-time simulation.

- 1 Open the model `sldrtex_model`.
- 2 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Mode > Run in Kernel**.
- 3 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Run in Real Time > Build**.

Simulink builds the real-time application.

- 4 On the **Desktop Real-Time** tab, click **Run in Real Time > Connect**.

Simulink connects the model and kernel.

- 5 On the **Desktop Real-Time** tab, click **Start**.

Simulink starts running the simulation.

- 6 To stop the simulation before it ends, on the **Desktop Real-Time** tab, click **Stop**.

These **Step by Step Commands** let you run a real-time simulation for a previously built real-time application.

- 1 If continuing from the previous procedure, close the model `sldrtex_model`.

- 2 Open the model `sldrtex_model`.

- 3 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Mode > Run in Kernel**.

- 4 On the **Desktop Real-Time** tab, click **Run in Real Time > Connect**.

Simulink connects the model and kernel.

- 5 On the **Desktop Real-Time** tab, click **Start**.

Simulink starts running the simulation.

- 6 To stop the simulation before it ends, on the **Desktop Real-Time** tab, click **Stop**.

See Also

More About

- “Execute Real-Time Application in Run in Kernel Mode by Using Run in Real Time” on page 3-24
- “Simulate Model in Connected IO Mode” on page 3-15
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Execute Real-Time Application with S-Functions in Run in Kernel Mode

You can use S-functions in **Run in Kernel** mode when the S-functions are self-contained. For example, the S-functions only call a limited subset of the standard C library, such as math and string functions and do not call file I/O, console I/O, or Windows API functions. An example of not-self-contained S-function would be a function that needs to `#include <windows.h>`.

Note that being able to simulate an S-function with other code generation targets—such as `grt.tlc`, `ert.tlc`, or `slrealtime.tlc`—does not prove that it can be simulated with `sldrt.tlc`. For example, S-functions that are linked with dynamically linked libraries (DLLs) on Windows generally do not simulate with Simulink Desktop Real-Time because the DLLs are not available to the real-time kernel.

In particular, the following functions are not supported in the C source code:

- File I/O (`fopen` and others)
- Process management (`spawn`, `exit`, and others)
- Signals and exceptions (`signal`, `longjmp`, `__try` and others)
- Time functions (`clock` and others)
- Any functions from the Windows API

The following C library functions are supported for use with Simulink Desktop Real-Time:

- Data conversion functions: `abs`, `atof`, `atoi`, `atol`, `itoa`, `labs`, `ltoa`, `strtod`, `strtol`, `strtoul`, `ultoa`,
- Memory allocation functions: `calloc`, `free`, `malloc`,
- Memory manipulation functions: `_memccpy`, `memcpy`, `memchr`, `memcmp`, `_memicmp`, `memmove`, `memset`,
- String manipulation functions: `strcat`, `strchr`, `strcmp`, `strcpy`, `strcspn`, `_strdup`, `_stricmp`, `strlen`, `_strlwr`, `strncat`, `strncmp`, `strncpy`, `_strnset`, `strpbrk`, `strrchr`, `_strrev`, `_strset`, `strspn`, `strstr`, `strtok`, `_strupr`,
- Mathematical functions: `acos`, `asin`, `atan`, `atan2`, `ceil`, `cos`, `cosh`, `div`, `exp`, `fabs`, `floor`, `fmod`, `frexp`, `ldexp`, `ldiv`, `log`, `log10`, `max`, `min`, `modf`, `pow`, `rand`, `sin`, `sinh`, `sqrt`, `srand`, `tan`, `tanh`, `uldiv`,
- Character class tests and conversion: `isalnum`, `isalpha`, `_isascii`, `isctrl`, `isdigit`, `isgraph`, `islower`, `isprint`, `ispunct`, `isspace`, `isupper`, `isxdigit`, `isxupper`, `isxlower`, `_toascii`, `tolower`, `toupper`,
- Searching and sorting: `bsearch`, `qsort`,
- Dummy functions - (can be there but do nothing) `exit`,
- Console I/O: `fprintf`, `printf`,

See Also

More About

- “Real-Time Execution in Run in Kernel Mode” on page 1-5

- “Custom I/O Driver Basics” on page A-2
- “Software Components” on page 2-2

Run Application from MATLAB Command Line

You can use the MATLAB command-line interface as an alternative to using the Simulink UI. Enter commands directly in the MATLAB Command Window or save them in a script file.

After you build the real-time application, you can run your model in real time.

This procedure uses the model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...  
    'sldrt','examples','sldrtex_model')))
```

It assumes that you have already created a real-time application from that model.

Connected IO Mode (Normal Mode)

- 1 In the MATLAB Command Window, type:

```
set_param(gcs, 'SimulationMode', 'normal')
```
- 2 To start running the simulation, type:

```
set_param(gcs, 'SimulationCommand', 'start')
```
- 3 To stop running the simulation, type:

```
set_param(gcs, 'SimulationCommand', 'stop')
```

Accelerator Mode

- 1 In the MATLAB Command Window, type:

```
set_param(gcs, 'SimulationMode', 'accelerator')
```
- 2 To start running the simulation, type:

```
set_param(gcs, 'SimulationCommand', 'start')
```
- 3 To stop running the simulation, type:

```
set_param(gcs, 'SimulationCommand', 'stop')
```

Run in Kernel Mode (External Mode)

- 1 In the MATLAB Command Window, type:

```
set_param(gcs, 'SimulationMode', 'external')
```
- 2 To load the real-time application and connect it to the Simulink block diagram, type:

```
set_param(gcs, 'SimulationCommand', 'connect')
```

```
Model sldrtex_model loaded
```
- 3 To start running the real-time application, type:

```
set_param(gcs, 'SimulationCommand', 'start')
```

- 4 To stop the real-time application, type:

```
set_param(gcs, 'SimulationCommand', 'stop')
```

See Also

More About

- “Real-Time Execution in Connected IO Mode” on page 1-3
- “Real-Time Execution in Run in Kernel Mode” on page 1-5

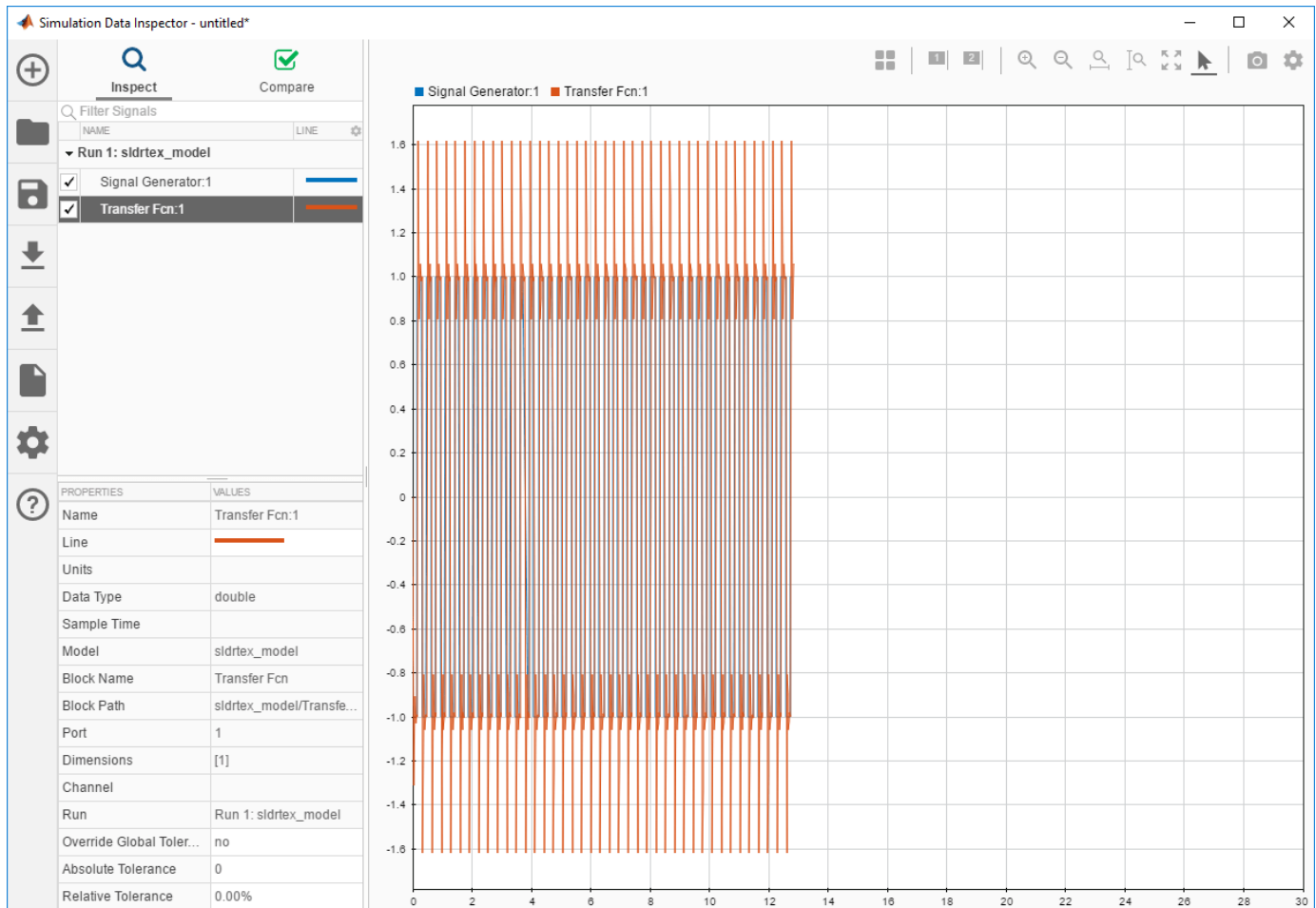
Inspect Simulink® Desktop Real-Time™ Signals with Simulation Data Inspector

This example shows how to use the Simulation Data Inspector (SDI) to log signal data from the real-time application. Use Simulink® **Run in Kernel** mode to establish a communication channel between your Simulink® block diagram and your real-time application. Control which signals to display by selecting them in the model. You can log signal data from models referenced at arbitrary levels within a model hierarchy.

This example uses the model `sldrtext_model`. To open this example, in the MATLAB Command Window, type:

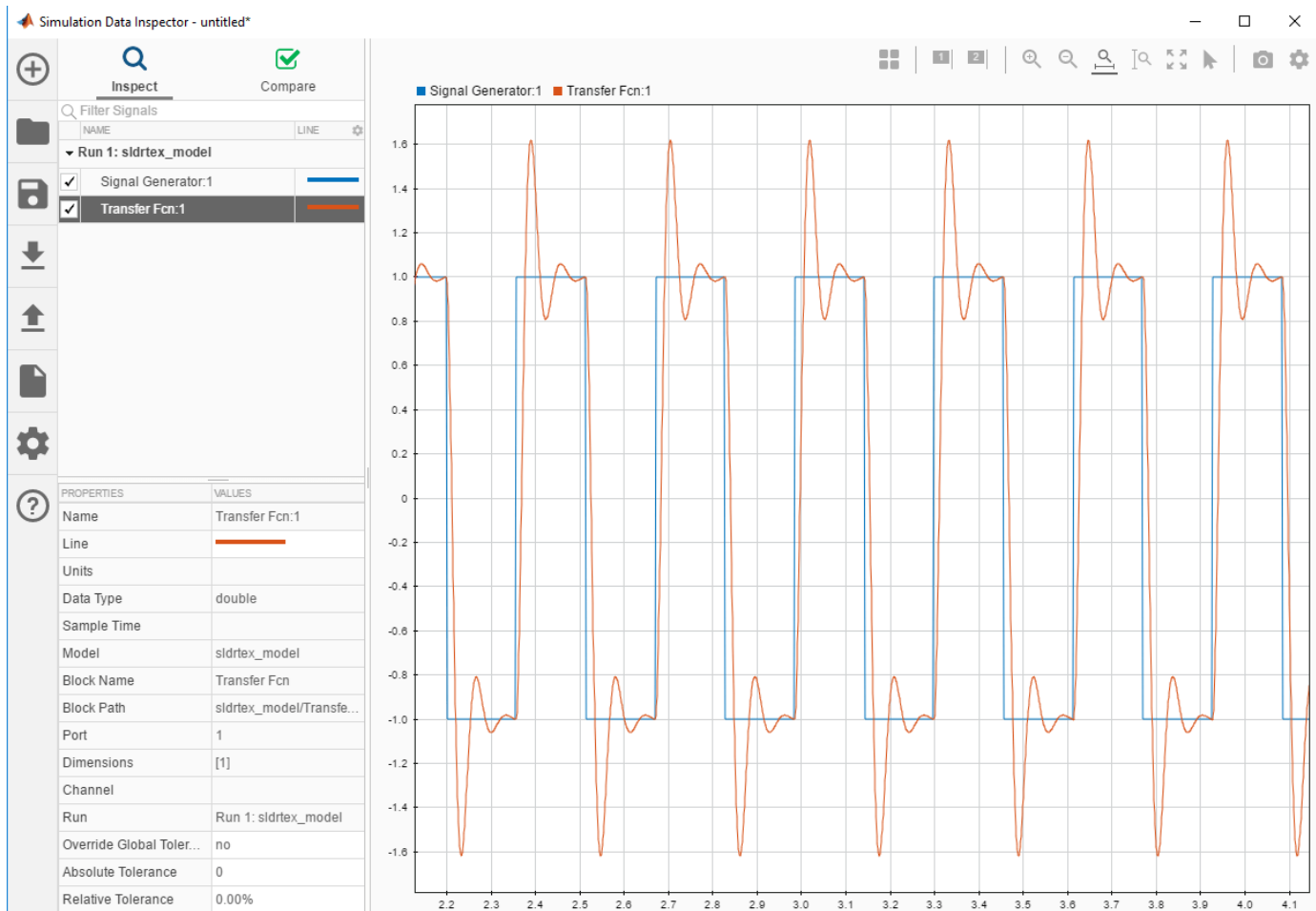
```
open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples',  
'sldrtext_model')))
```

- 1 Open `sldrtext_model`.
- 2 In the Simulink Editor, on the **Simulation** tab, set the simulation stop time to, for example, 30 seconds.
- 3 In the model, select the signals `Signal Generator` and `Transfer Fcn`.
- 4 On the **Desktop Real-Time** tab, click **Log Signals**. A faint Simulation Data Inspector icon appears next to each signal.
- 5 To start the real-time execution in **Run in Kernel** mode, on the **Desktop Real-Time** tab, click **Run in Real Time**. The Simulation Data Inspector button glows, indicating that Simulation Data Inspector has data available for viewing.
- 6 On the **Desktop Real-Time** tab, click **Data Inspector**.
- 7 In the Simulation Data Inspector, select the signals `SignalGenerator:1` and `Transfer Fcn:1`. Simulation Data Inspector displays plotted data.



10. To stop the simulation, click the **Stop** button.

11. After the simulation, use the toolbar buttons to explore the data. For example, to view the simulation between seconds 2 and 4, in Simulation Data Inspector, click the **Zoom in Time** button. Drag the cursor over the range from 2 to 4.



12. To save the Simulation Data Inspector session as a `.mat` file, click **Save**.

See Also

Simulation Data Inspector

More About

- “Water Tank Model with Dashboard” on page 5-5
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Signal Logging to the Workspace

Logging signals to the workspace saves data to a variable in your MATLAB base workspace. You can use MATLAB functions for data analysis and MATLAB plotting functions for visualization. You can save data to a variable during a simulation or during an execution.

The steps in this process are:

- 1 “Set Scope Parameters for Logging to Workspace” on page 3-36
- 2 “Set Run in Kernel Mode Properties for Logging to Workspace” on page 3-38
- 3 “Plot Signal Data Logged to Workspace” on page 3-40

If your model contains **Outport** blocks, you cannot save signal data in **Run in Kernel** mode. Simulink supports signal logging with **Outport** blocks in **Connected IO** mode or accelerator modes only, when Simulink runs the simulation algorithm.

Tip In **Run in Kernel** mode, do not enter or select parameters on the **Data I/O** tab in the Configuration Parameters dialog box. Instead, add a **Scope** block to your Simulink model to log signal data.

See Also

More About

- “Signal Logging to a File” on page 3-42


Set Scope Parameters for Logging to Workspace

Data is saved to the MATLAB workspace through a Simulink Scope block. For data to be saved, set Scope block parameters. After you create a Simulink model and add a Scope block, you can enter the scope parameters for signal logging to the MATLAB workspace.

This procedure uses the model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

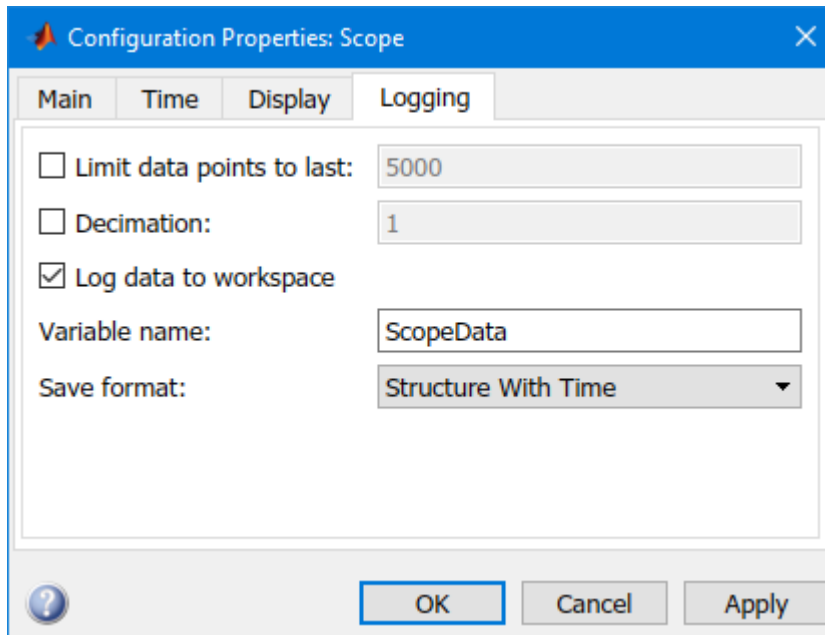
```
open_system(docpath(fullfile(docroot,'toolbox',...  
    'sldrt','examples','sldrtex_model')))
```

It assumes that you have already loaded that model.

- 1 In the Simulink Editor, on the Desktop Real-Time tab, select **Mode > Run in Kernel**.
- 2 In the model diagram, double-click the Scope block.
- 3 On the Scope toolbar, click the **Parameters** button .
- 4 In the Scope Parameters dialog box, click the **Logging** tab.
- 5 Do one of the following:
 - If you are running a normal mode simulation in **Connected IO** mode, select the **Limit data points to last** check box, and enter the number of sample points to save.
 - If you are running an external mode simulation in **Run in Kernel** mode, do not select the **Limit data points to last** check box.

When you are using Simulink Desktop Real-Time software, use the **Duration** value to set the number of sample points you save. To set the **Duration** value, see “Set Run in Kernel Mode Properties for Logging to Workspace” on page 3-38. For more information, see “External Mode Simulation with TCP/IP or Serial Communication” (Simulink Coder).

- 6 Select the **Log data to workspace** check box. In the **Variable name** text box, enter the name of a MATLAB variable. The default name is `ScopeData`.
- 7 From the **Save format** list, select one of `Structure with time`, `Structure`, `Array`, and `Dataset`. For example, to save the sample times and signal values at those times, select `Structure with time`.



- 8 Click **OK**.

When you modify a value in the Scope parameters dialog box, you must click the **Apply** or **OK** button for the changes to take effect. Rebuild your real-time application before connecting and starting it. If you do not rebuild, an error dialog box opens. If you do not click **Apply**, your executable runs, but it uses the old settings.

See Also

More About

- "Signal Logging to a File" on page 3-42
- "Troubleshoot Missing Desktop Real-Time Tab" on page 6-2

Set Run in Kernel Mode Properties for Logging to Workspace

Data is saved to the MATLAB workspace through a Simulink Scope block. Set signal and triggering properties only when you are running a real-time application. If you are running a **Connected IO** mode or accelerator mode simulation, you can skip this procedure.

After you create a Simulink model and add a Scope block, you can enter the signal and triggering properties for logging to the MATLAB workspace.

This procedure uses the model `sldrtext_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot, 'toolbox', ...  
    'sldrt', 'examples', 'sldrtext_model')))
```

The steps in this procedure assume that you have already loaded that model and have completed the steps in “Set Scope Parameters for Logging to Workspace” on page 3-36.

- 1 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Prepare > Control Panel**.
- 2 In the External Mode Control Panel, click the **Signal & Triggering** button.
- 3 Click the **Select all** button. From the Source list, select `manual`. From the Mode list, select `normal`.

The X under **Signal selection** designates that a signal has been tagged for data collection, and T designates that the signal has been tagged as a trigger signal.

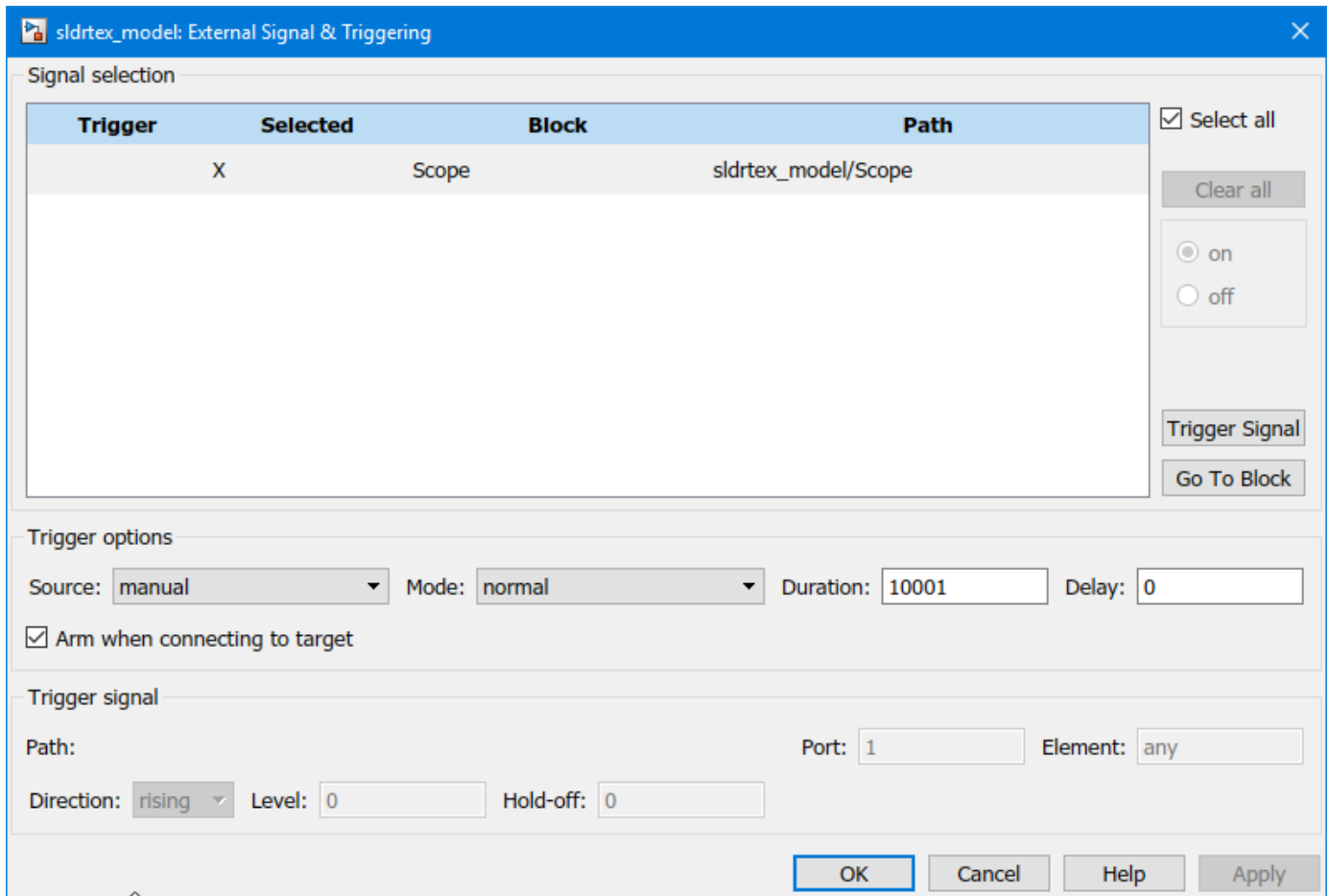
- 4 In the **Duration** field, enter the number of sample points in a data buffer. Enter a **Duration** value equal to the total number of sample points that you must collect for a run. For example, if you have a sample rate of 1000 samples/second and a stop time of 10 seconds, enter:

```
10001
```

Set the time axis for Simulink Scope blocks equal to the sample interval (in seconds) times the number of points in each data buffer. This setting displays one buffer of data across the entire Simulink Scope plot.

- 5 Clear the **Limit data points to last** check box. See “Set Scope Parameters for Logging to Workspace” on page 3-36.

For more information, see “External Mode Simulation with TCP/IP or Serial Communication” (Simulink Coder).



6 Click **OK**.

In the External Signal & Triggering dialog box, click the **OK** button for the changes you made to take effect. You do not have to rebuild your real-time application.

See Also

More About

- “Signal Logging to a File” on page 3-42
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Plot Signal Data Logged to Workspace

To visualize non-real-time simulated data or real-time application data, use the MATLAB plotting functions.

After running your real-time application and logging data to the MATLAB workspace, you can plot the data.

This procedure uses the model `sldrtext_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot, 'toolbox', ...
    'sldrt', 'examples', 'sldrtext_model')))
```

It assumes that you saved your data to the variable `ScopeData`.

The steps in this procedure assume that you have already loaded that model and have completed the steps in “Set Scope Parameters for Logging to Workspace” on page 3-36 and “Set Run in Kernel Mode Properties for Logging to Workspace” on page 3-38.

- 1 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Run in Real Time**.

When the real-time application runs, it creates the `ScopeData` variable that you configured in “Set Run in Kernel Mode Properties for Logging to Workspace” on page 3-38.

- 2 To show the structure of the variable `ScopeData`, in the MATLAB Command Window, type:

```
ScopeData
ScopeData =
    struct with fields:
        time: [2001x1 double]
        signals: [1x1 struct]
        blockName: 'sldrtext_model/Scope'
```

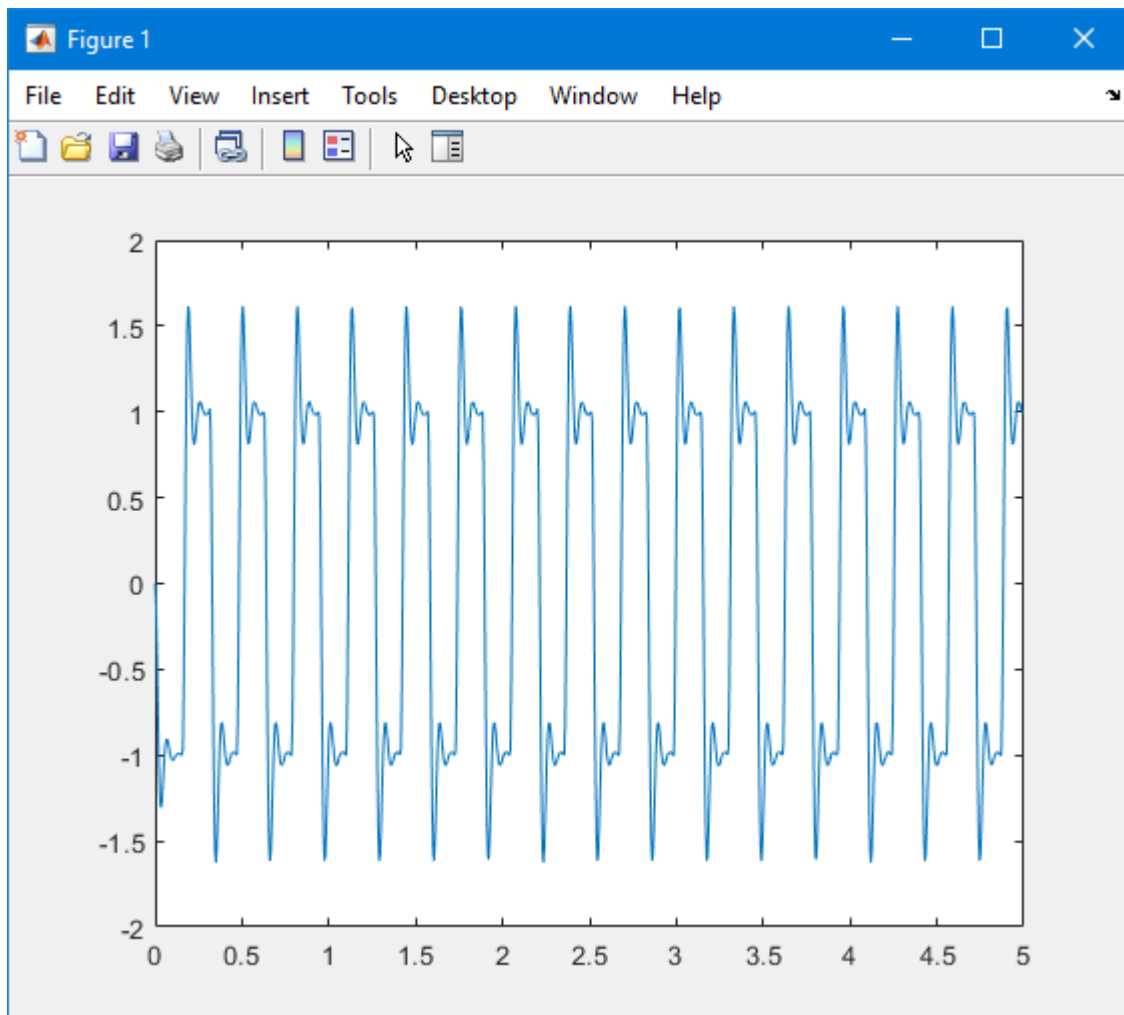
To list the contents of the structure `signals`, type:

```
ScopeData.signals
ans =
    struct with fields:
        values: [2001x1 double]
        dimensions: 1
        label: ''
        title: ''
        plotStyle: 0
```

- 3 To plot the first 1000 points, type:

```
plot(ScopeData.time(1:1000), ScopeData.signals.values(1:1000))
```

The MATLAB environment plots the first 1000 samples over 0.0000–0.9990 seconds.



- 4 The variable `ScopeData` is not automatically saved to your hard disk. To save the variable `ScopeData`, type:

```
save ScopeData
```

The MATLAB environment saves the scope data to the file `ScopeData.mat`.

See Also

More About

- “Signal Logging to a File” on page 3-42
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Signal Logging to a File

Logging signals to a file saves data to a variable in your MATLAB workspace, and then saves that data to a MAT-file. You can use MATLAB functions for data analysis and MATLAB plotting functions for visualization on the data in the MAT-file.

The steps in this process are:

- 1 “Set Scope Parameters for Logging to File” on page 3-43
- 2 “Set Run in Kernel Mode Properties for Logging to File” on page 3-45
- 3 “Set Run in Kernel Mode Data Archiving Parameters” on page 3-47
- 4 “Plot Signal Data Logged to File” on page 3-49

If your model contains `Outport` blocks, you cannot save signal data in **Run in Kernel** mode. Simulink supports signal logging to a file in **Connected IO** mode or accelerator mode only, when Simulink runs the simulation algorithm.

Tip In **Run in Kernel** mode, do not enter or select parameters on the **Data I/O** tab in the Configuration Parameters dialog box. Instead, add a `Scope` block to your Simulink model and use it to log signal data for data archiving.

See Also

More About

- “Signal Logging to the Workspace” on page 3-35

Set Scope Parameters for Logging to File


You save data to a file by first saving the data to the MATLAB workspace through a Simulink Scope block. For data to be saved, set Scope block parameters.

After you create a Simulink model and add a Scope block, you can enter the scope parameters for signal logging to a file.

This procedure uses the model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot, 'toolbox', ...
    'sldrt', 'examples', 'sldrtex_model')))
```

It assumes that you have already loaded that model.

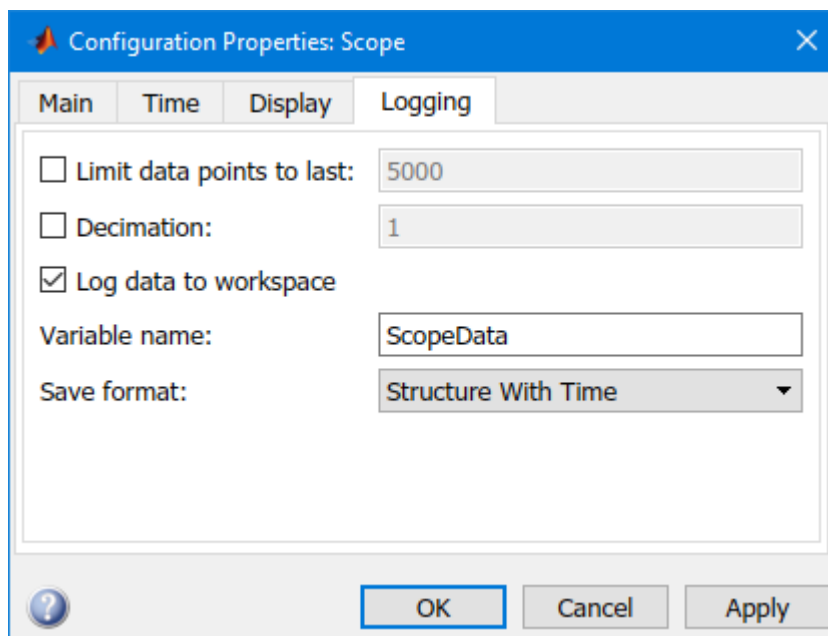
- 1 In the Simulink Editor, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 2 In the model diagram, double-click the Scope block.
- 3 On the **Scope** toolbar, click the **Parameters** button .
- 4 Click the **Logging** tab.
- 5 Do one of the following:
 - If you are running a normal mode simulation in **Connected IO** mode, select the **Limit data points to last** check box, and enter the number of sample points to save.
 - If you are running an external mode simulation in **Run in Kernel** mode, do not select the **Limit data points to last** check box.

When you are using Simulink Desktop Real-Time software, use the **Duration** value to set the number of sample points you save. To set the **Duration** value, see “Set Run in Kernel Mode Properties for Logging to File” on page 3-45. For more information, see “External Mode Simulation with TCP/IP or Serial Communication” (Simulink Coder).

- 6 Select the **Log data to workspace** check box. In the **Variable name** text box, enter the name of a MATLAB variable. The default name is `ScopeData`.

In the Scope parameters dialog box, you must select the **Log data to workspace** check box to be able to save data to a file. If you do not select the **Log data to workspace** check box, the MAT-files for data logging are created, but they are empty.

- 7 From the **Save format** list, select one of **Structure with time**, **Structure**, **Array**, and **Dataset**. For example, to save the sample times and signal values at those times, select **Structure with time**.



- 8 Click **OK**.

Before connecting and starting the application with changed settings, rebuild your real-time application. If you do not rebuild after these changes, an error occurs.

See Also

More About

- "Signal Logging to the Workspace" on page 3-35
- "Troubleshoot Missing Desktop Real-Time Tab" on page 6-2

Set Run in Kernel Mode Properties for Logging to File

Data is saved to a file by first saving the data to the MATLAB workspace through a Simulink Scope block. Before running a real-time application, set signal and triggering properties.

After you create a Simulink model and add a Scope block, you can enter the signal and triggering properties for data logging to a file.

This procedure uses the model `sldrtext_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot, 'toolbox', ...
    'sldrt', 'examples', 'sldrtext_model')))
```

The steps in this procedure assume that you have already loaded that model and have completed the steps in “Set Scope Parameters for Logging to File” on page 3-43.

- 1 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Prepare > Control Panel**.
- 2 In the External Mode Control Panel, click the **Signal & Triggering** button.
- 3 Click the **Select all** check box. From the **Source** list, select `manual`. From the **Mode** list, select `normal`.

The X under **Signal selection** designates that a signal has been tagged for data collection, and T designates that the signal has been tagged as a trigger signal.

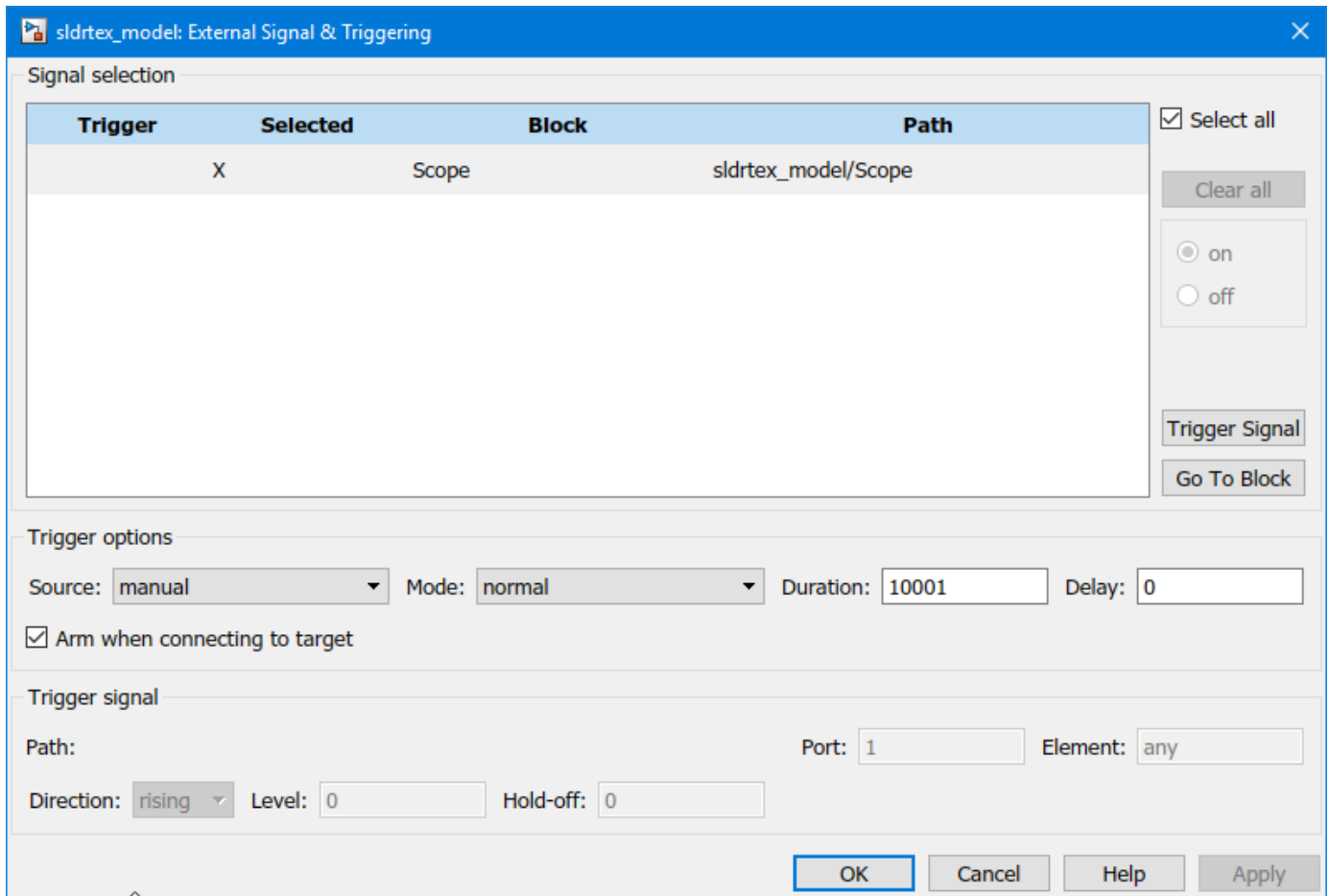
- 4 In the **Duration** field, enter the number of sample points in a data buffer. Enter a **Duration** value equal to the total number of sample points that you must collect for a run. For example, if you have a sample rate of 1000 samples/second and a stop time of 10 seconds, enter:

```
10001
```

Set the time axis for Simulink Scope blocks equal to the sample interval (in seconds) times the number of points in each data buffer. This setting displays one buffer of data across the entire Simulink Scope plot.

- 5 Clear the **Limit data points to last** check box. See “Set Scope Parameters for Logging to Workspace” on page 3-36.

For more information, see “External Mode Simulation with TCP/IP or Serial Communication” (Simulink Coder).



- 6 Click **OK**.

In the External Signal & Triggering dialog box, click the **OK** button for the changes you made to take effect. You do not have to rebuild your real-time application.

See Also

More About

- “Signal Logging to the Workspace” on page 3-35
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Set Run in Kernel Mode Data Archiving Parameters

You must select the **Log data to workspace** check box in the Scope parameters dialog box for the software to save data in the data logging MAT-files.

This procedure uses the model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot, 'toolbox', ...  
    'sldrt', 'examples', 'sldrtex_model')))
```

The steps in this procedure assume that you have already loaded that model and have completed the steps in “Set Scope Parameters for Logging to File” on page 3-43 and “Set Run in Kernel Mode Properties for Logging to File” on page 3-45.

After you create a Simulink model, you can enter the Data Archiving Parameters for data logging to a file:

- 1 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Prepare > Control Panel**.
- 2 In the External Mode Control Panel, click the **Data Archiving** button.
- 3 Select the **Enable archiving** check box.
- 4 In the **Directory** text box, enter the path to a folder on your disk. For example, if your MATLAB working folder is named `mwd`, enter

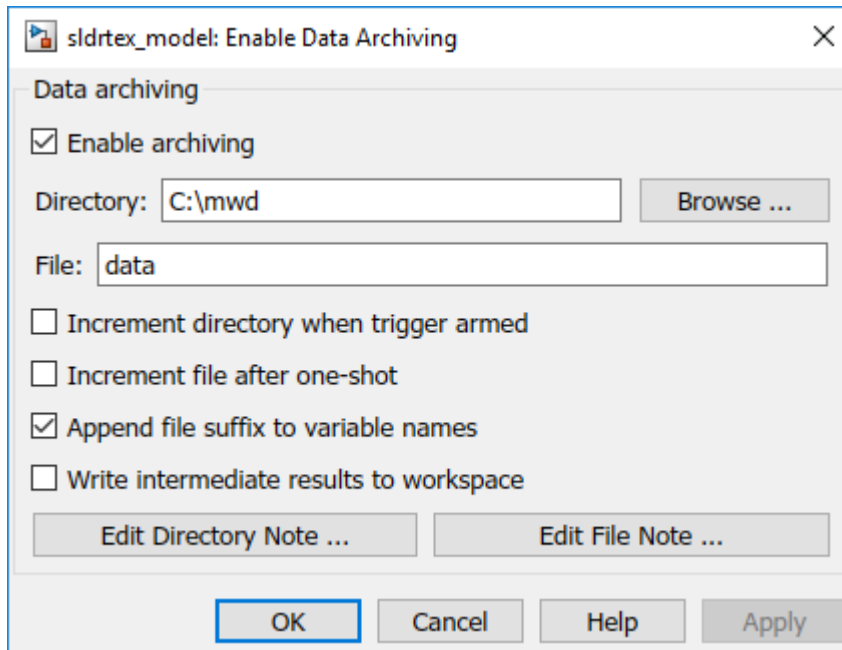
```
c:\mwd
```

- 5 In the **File** text box, enter the file name prefix for the data files to be saved. For example, enter:

```
data
```

The MATLAB environment names the files `data_0.mat`, `data_1.mat`, and so on. The number of files equals the total sample points. For example, if you set **Duration** to `Total sample points`, then only one file is created.

- 6 Select the **Append file suffix to variable names** check box.



- 7 Click the **OK** button.

In the External Signal & Triggering dialog box, click the **OK** button for the changes you made to take effect. You do not have to rebuild your real-time application.

See Also

More About

- “Signal Logging to the Workspace” on page 3-35
- “Signal Logging to a File” on page 3-42
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Plot Signal Data Logged to File

You can use the MATLAB plotting functions for visualization of your non-real-time simulated data or your real-time executed data.

After running your real-time application and logging data to a file, you can plot the data.

This procedure uses the model `sldrtext_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot, 'toolbox', ...
    'sldrt', 'examples', 'sldrtext_model')))
```

It assumes that you saved your data to the variable `ScopeData`.

The steps in this procedure assume that you have already loaded that model and have completed the steps in “Set Scope Parameters for Logging to File” on page 3-43, “Set Run in Kernel Mode Properties for Logging to File” on page 3-45, and “Set Run in Kernel Mode Data Archiving Parameters” on page 3-47.

- 1 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Run in Real Time**.

When the real-time application runs, it creates the `ScopeData` variable that you configured in “Set Run in Kernel Mode Properties for Logging to File” on page 3-45 and creates the archive data files that you configured in “Set Run in Kernel Mode Data Archiving Parameters” on page 3-47.

- 2 In the MATLAB Command Window, type:

```
ScopeData

ScopeData =
    time: [10000x1 double]
    signals: [1x1 struct]
    blockName: 'sldrtext_model/Scope'
```

- 3 To list the MAT-files saved to your disk, type:

```
dir *.mat

data_0.mat
```

- 4 To clear the MATLAB workspace and load the scope data, type:

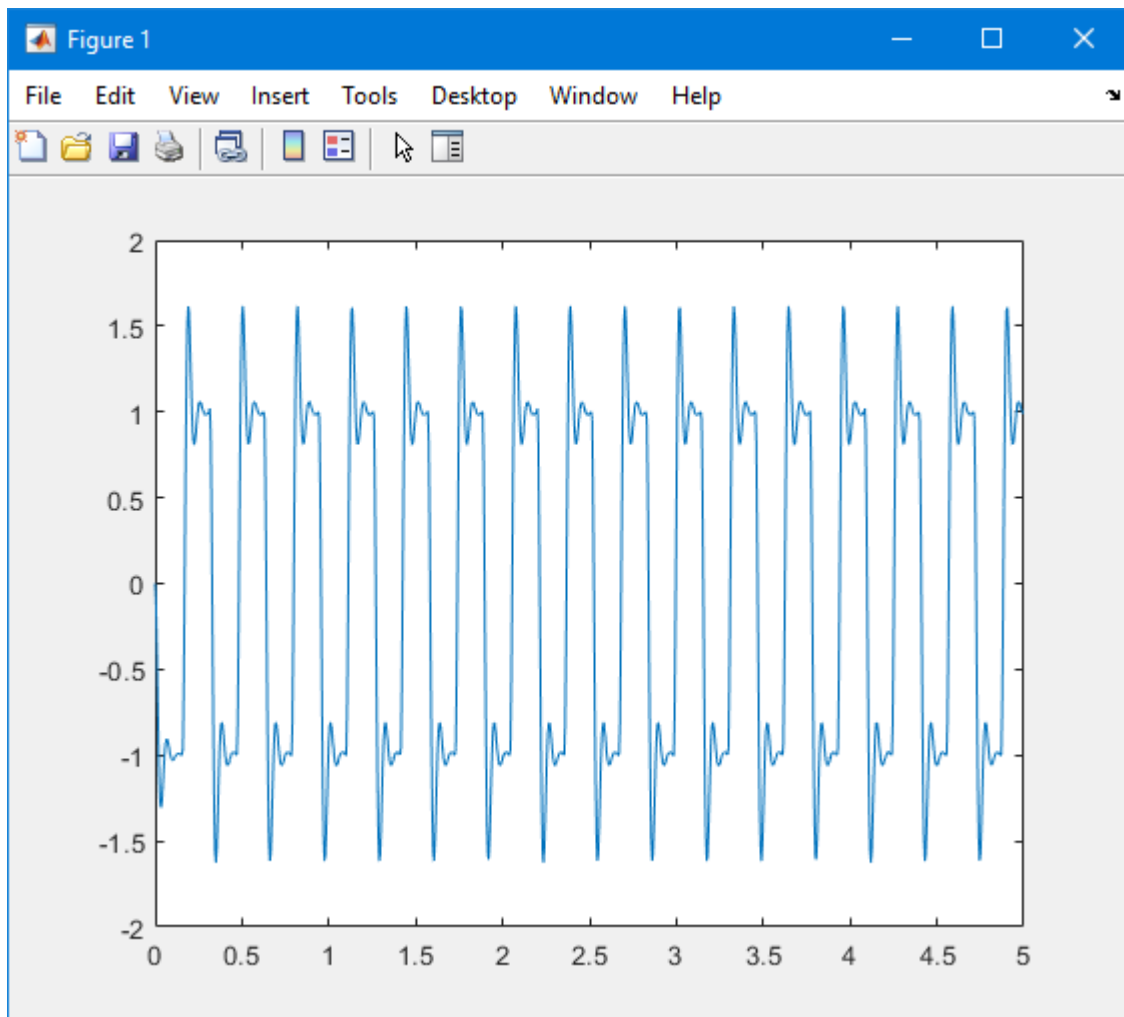
```
clear
load data_0
who

Your variables are:
ScopeData
```

- 5 To plot the first 1000 points, type:

```
plot(ScopeData_0.time(1:1000), ...
    ScopeData_0.signals.values(1:1000))
```

The MATLAB environment plots the first 1000 samples over 0.0000–0.9990 seconds.



See Also

More About

- "Signal Logging to the Workspace" on page 3-35
- "Signal Logging to a File" on page 3-42
- "Troubleshoot Missing Desktop Real-Time Tab" on page 6-2

Tunable Block Parameters and Tunable Global Parameters

To change the behavior of a model, you can tune Simulink Desktop Real-Time block parameters, provided the parameters are tunable. You can change block parameters via the block parameter dialog box, Dashboard blocks, and MATLAB language. You can create tunable global parameters by using MATLAB variables as value expressions.

In **Connected IO** mode or accelerator mode, Simulink transfers the new values to the model that is being simulated. In **Run in Kernel** mode, Simulink transfers the new values to the real-time application that is running in the kernel mode process.

Procedures to consider when working with tunable parameters include:

- 1 “Tune Block Parameters by Using the Block Dialog Box” on page 3-54
- 2 “Tune Block Parameters with Data Navigation” on page 3-57
- 3 “Sweep MATLAB Variables with MATLAB Scripting” on page 3-62

Tunable Parameters

Simulink Desktop Real-Time defines two kinds of tunable parameters: block parameters and global parameters.

Tunable Block Parameters

A block parameter is a constant expression that you reference in a Simulink block dialog box or by using the MATLAB API. Block parameters are tunable when you set the **Default parameter behavior** option to **Tunable** on the **Optimization** pane. When using the MATLAB API, you identify a block parameter by the parameter name and the block path in the model hierarchy.

Suppose that you set the **Amplitude** parameter of a Signal Generator block to a value of 5/2. You can change the amplitude of the signal generator during simulation by tuning parameter **Amplitude** in block **Signal Generator**.

Tunable Global Parameter

A tunable global parameter is a MATLAB variable that you reference in a Simulink block dialog box. You can tune a global parameter or object by using a block dialog box, Dashboard blocks, Property Inspector, Model Explorer, Model Data Editor, or MATLAB language. When using the MATLAB API, you identify a tunable global parameter by the variable name only.

Suppose that you assign to the **Amplitude** parameter the variable **A** with the value 4.57. You can change the amplitude of the signal generator during simulation by tuning the value of **A** in the MATLAB workspace and updating the simulation.

Inlined Parameters

To improve execution efficiency, open the Configuration Parameters dialog box and set the **Default parameter behavior** option to **Inlined** on the **Code Generation > Optimization** pane.

By default, you cannot tune inlined block parameters. However, you can create a tunable global variable by referencing a MATLAB variable or `Simulink.Parameter` object in the block dialog box. To make the variable or object tunable, apply a storage class other than `Auto` to it.

For more information about inlined parameters, see “Default parameter behavior” (Simulink Coder).

Tune Parameters by Using Run in Kernel Mode

In **Run in Kernel** mode, Simulink Desktop Real-Time connects your Simulink model to your real-time application. The block diagram becomes a user interface for the real-time application. You can change a parameter value in a block dialog box or replace the value with a MATLAB variable and tune the variable in the Command Window.

When you change a parameter value in a Simulink model and click **OK**, Simulink Desktop Real-Time transfers the data to the real-time application and changes the block parameter. You can change only the parameters that do not change the model structure. If you modify the structure, you must recompile the model.

If you change the value of a tunable global parameter, instruct Simulink to transfer the data from the MATLAB variable to the real-time application by either:

- Pressing **Ctrl+D**.
- In the Simulink Editor, on the **Debug** tab, clicking **Update Model**.

Tune Parameters by Using Hold Updates and Update All Parameters

By using **Hold Updates**, you can tune multiple parameters and apply all of the tuned parameters at once, instead of tuning one parameter at a time. This example uses model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...
    'sldrt','examples','sldrtex_model')))
```

- 1 Open model `sldrtex_model`.
- 2 In the Simulink Editor, on the **Desktop Real-Time** tab, click **Run in Real Time**.
- 3 On the **Desktop Real-Time** tab, click **Prepare > Hold Updates**. The editor remains in **Hold Updates** mode until you click **Hold Updates** again.

To set parameter values, you can set values either by clicking each block or by using the Model Data Editor in the base workspace.

- 4 On the **Desktop Real-Time** tab, **Prepare > Signal Table**.
- 5 In the Model Data Editor, click the **Parameters** tab. Modify parameters values in the Model Data Editor in the base workspace.
- 6 On the **Desktop Real-Time** tab, click **Prepare > Update All Parameters**.
- 7 To stop the simulation before it ends, on the **Desktop Real-Time** tab, click **Stop**.

Tune Parameters by Using the MATLAB Language

In Simulink Desktop Real-Time, you can use the MATLAB language command `set_param` to change the values of block parameters and tunable global parameters. This example uses model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...
    'sldrt','examples','sldrtex_model')))
```


If you are using a literal block parameter value, you access the parameter by a nonempty block path and the parameter name. For example, to change the amplitude of the signal generator:

```
model = 'sldrtex_model';  
sgname = [model '/Signal Generator'];  
set_param(sgname, 'Amplitude', '4.57')
```

If you are replacing a block parameter with a tunable global parameter, you access the parameter by variable name. Suppose that you set **Amplitude** to the variable A. To change the amplitude of the signal generator:

```
A = 4.57  
set_param('sldrtex_model', 'SimulationCommand', 'update')
```

See Also

More About

- “Tune Block Parameters by Using the Block Dialog Box” on page 3-54
- “Tune Block Parameters with Data Navigation” on page 3-57
- “Sweep MATLAB Variables with MATLAB Scripting” on page 3-62
- “Water Tank Model with Dashboard” on page 5-5
- “Default parameter behavior” (Simulink Coder)
- “Tune and Experiment with Block Parameter Values”
- “Share and Reuse Block Parameter Values by Creating Variables”
- “How Generated Code Stores Internal Signal, State, and Parameter Data” (Simulink Coder)
- “Preserve Variables in Generated Code” (Simulink Coder)
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Tune Block Parameters by Using the Block Dialog Box

After running your real-time application, use the block mask dialog box or Property Inspector to change parameter values and observe the changes to the signals. In **Connected IO** mode or accelerator mode, Simulink transfers the new values to the model that is being simulated. In **Run in Kernel** mode, Simulink transfers the new values to the real-time application that is running in the kernel mode process.

For this example, your goal is to minimize ringing in the transfer function.

This procedure begins with the square-wave transfer function model `sldrtex_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...
    'sldrt','examples','sldrtex_model')))
```

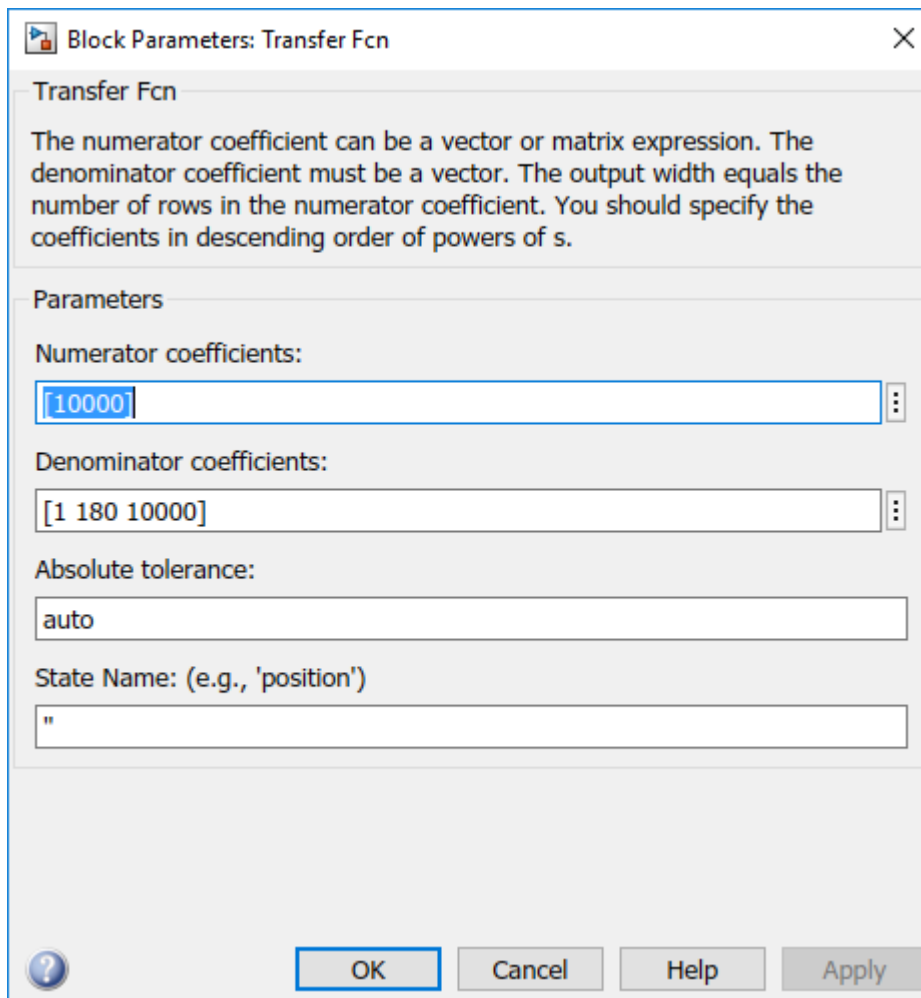
This model opens in **Connected IO** mode.

First, install the Simulink Desktop Real-Time kernel and cd to a working folder.

- 1 Open `sldrtex_model`.
- 2 Open the Scope block.
- 3 In Simulink Editor, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 4 On the **Desktop Real-Time** tab, select **Run in Real Time > Stop Time** and change the **Stop Time** to Inf.
- 5 On the **Desktop Real-Time** tab, click **Run in Real Time**. Simulink builds the real-time application, connects to the real-time kernel, and starts running the real-time simulation.



- 6 Open the Transfer Fcn block parameters dialog box.
- 7 Change **Denominator coefficients** to [1 180 10000].



The image shows a dialog box titled "Block Parameters: Transfer Fcn". It contains a section for "Transfer Fcn" with explanatory text, a "Parameters" section with input fields for "Numerator coefficients" (containing [10000]), "Denominator coefficients" (containing [1 180 10000]), "Absolute tolerance" (containing auto), and "State Name" (containing "). At the bottom are buttons for "?", "OK", "Cancel", "Help", and "Apply".

Block Parameters: Transfer Fcn

Transfer Fcn

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s .

Parameters

Numerator coefficients:
[10000]

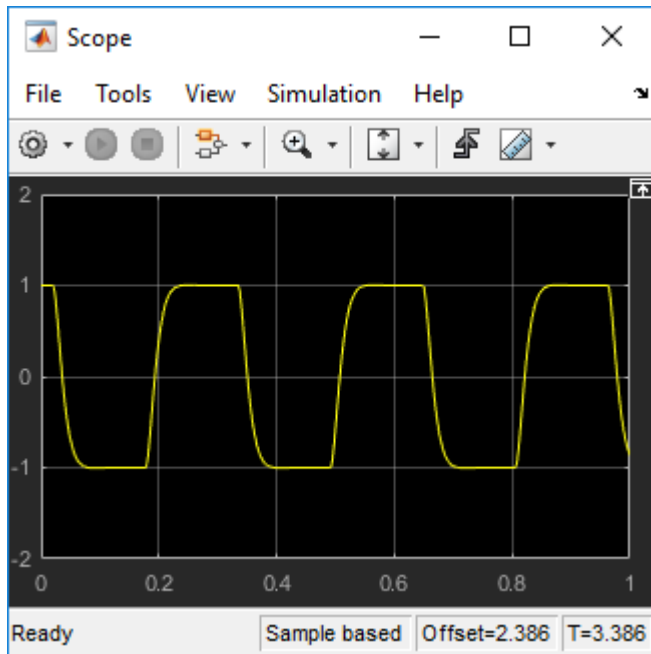
Denominator coefficients:
[1 180 10000]

Absolute tolerance:
auto

State Name: (e.g., 'position')
"

? OK Cancel Help Apply

- 8 Click **Apply**.



- 9 On the **Desktop Real-Time** tab, click **Stop**.

See Also

More About

- “Tunable Block Parameters and Tunable Global Parameters” on page 3-51
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Tune Block Parameters with Data Navigation

You can create tunable global parameters by embedding MATLAB variables in block dialog boxes with data navigation. You can tune the parameters by changing the variable values during execution. In **Connected IO** mode or accelerator mode, Simulink transfers the new values to the model that is being simulated. In **Run in Kernel** mode, Simulink transfers the new values to the real-time application that is running in the kernel mode process.

You can permanently store parameter objects and other external data in a data dictionary.

For this example, your goal is to minimize ringing in the transfer function.

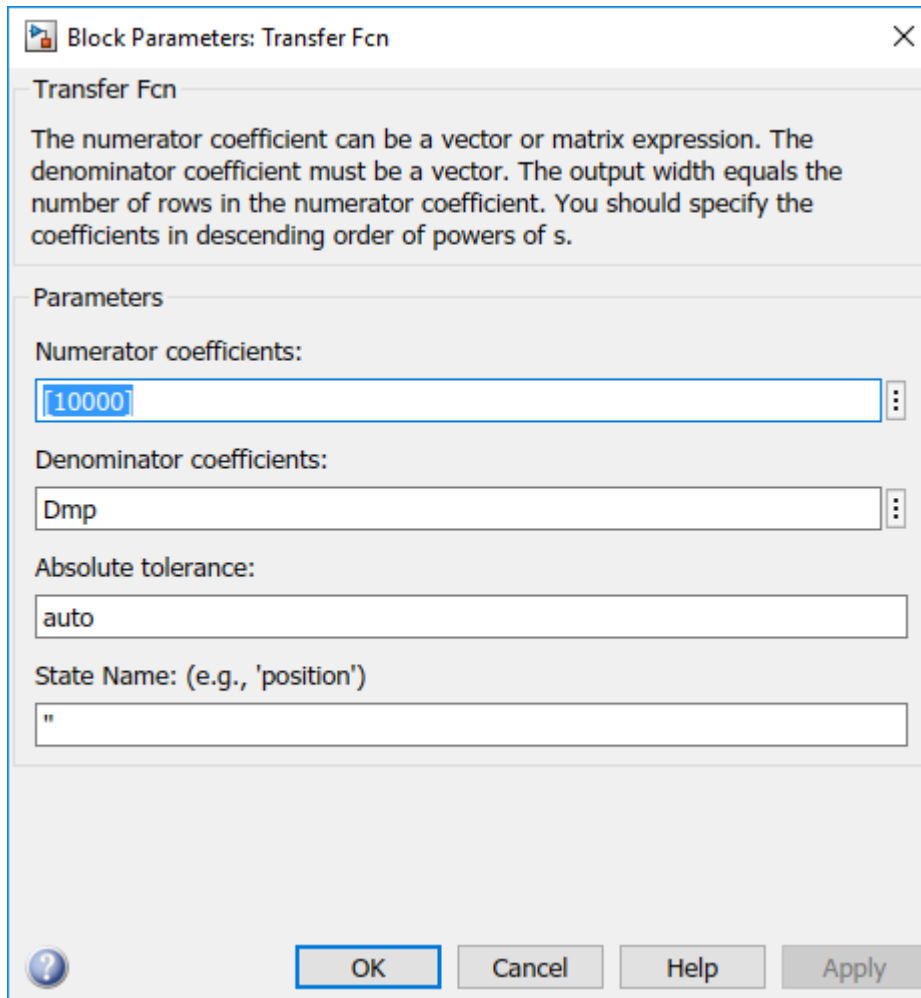
This procedure begins with the square-wave transfer function model `sldrtext_model`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot,'toolbox',...  
    'sldrt','examples','sldrtext_model')))
```

This model opens in **Connected IO** mode.

Create Parameter Object

- 1 Change to a working folder by using the `cd` command.
- 2 Open `sldrtext_model`.
- 3 Open the Transfer Fcn block parameters dialog box.
- 4 Replace the existing value of **Denominator coefficients** (`[1 70 10000]`) with `Dmp`.



- 5 Click the **Property Actions** \ddots button after Dmp and select **Dmp: Create**.
- 6 In the **Value** field, select Simulink.Parameter.
- 7 In the **Location** field, select **Base Workspace**.
- 8 Click **Create**.

If the model is already in **Run in Kernel** mode, the data type defaults to Simulink.Parameter in the base workspace.


- 9 In the **Simulink.Parameter: Dmp** dialog box, in the **Value** field, enter [1 70 10000].

For the rest of the fields, take the defaults.

- 10 In the **Simulink.Parameter: DMP** dialog box, click **Apply** and then click **OK**.
- 11 In the **Block Parameters: Transfer Fcn** dialog box, click **OK**.

Tune Parameter Object

This procedure continues from the steps in “Create Parameter Object” on page 3-57.

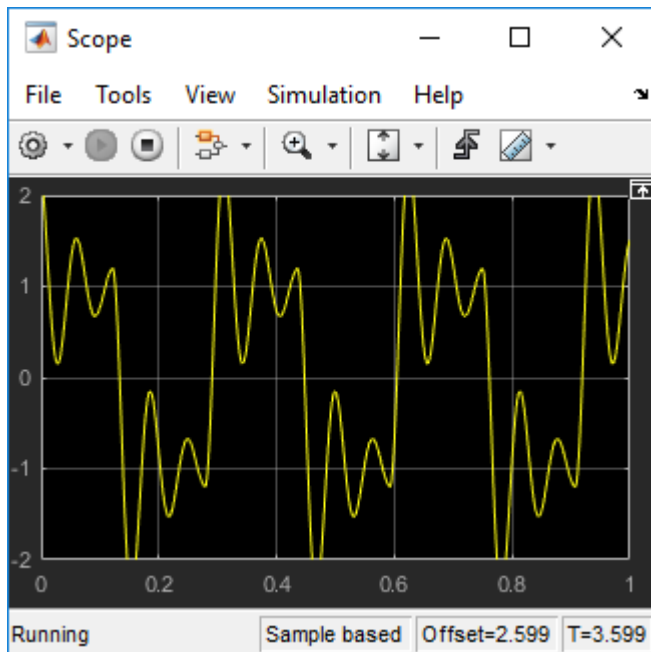
- 1 In the Simulink Editor, on the **Desktop Real-Time** tab, select **Run in Real Time > Stop Time** and change the **Stop Time** to **Inf**.
- 2 Open the Scope block.
- 3 Open the Transfer Fcn block parameters dialog box.
- 4 Click the **Property Actions**  button after Dmp and select **Dmp (base workspace) > Open**.

Before you start execution, open this dialog box. You cannot open variable Dmp while the real-time application is running.

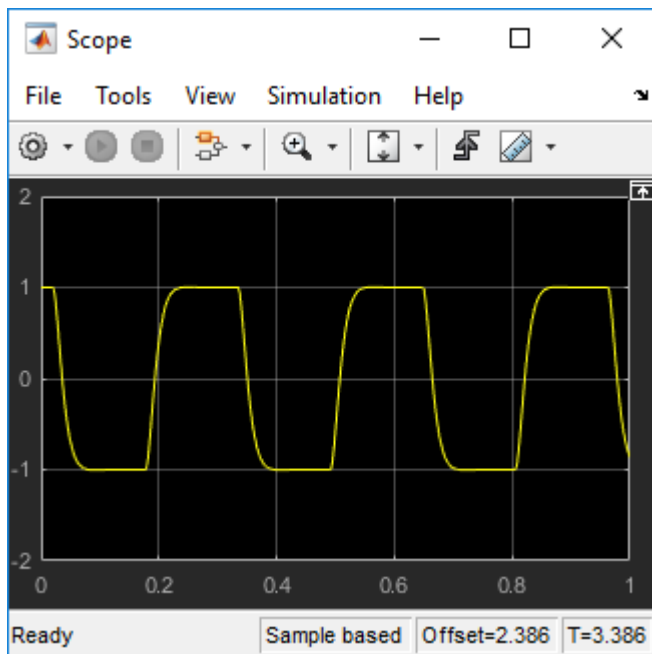
- 5 To start execution, on the **Desktop Real-Time** tab, click **Run in Real Time**.



- 6 In the **Simulink.Parameter: Dmp** dialog box, change **Value** to [1 30 10000] and click **Apply**.
- 7 Change the active dialog box by clicking on the model in the Simulink Editor, and then press **Ctrl-D**.



- 8 Change **Value** to [1 180 10000] and click **Apply**.
- 9 Change the active dialog box by clicking on the model in the Simulink Editor, and then press **Ctrl-D**.



10 On the **Desktop Real-Time** tab, click **Stop**.

See Also

More About

- “Share and Reuse Block Parameter Values by Creating Variables”
- “Tunable Block Parameters and Tunable Global Parameters” on page 3-51
- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2

Sweep MATLAB Variables with MATLAB Scripting

You can embed MATLAB variables in the base workspace with MATLAB commands and use MATLAB language to change their values during execution. In **Connected IO** mode or accelerator mode, Simulink transfers the new values to the model that is being simulated. In **Run in Kernel** mode, Simulink transfers the new values to the real-time application that is running in the kernel mode process.

For this example, your goal is to minimize ringing in the transfer function. For improved performance, you have inlined block parameters. When you inline block parameters, the parameters appear as nontunable constants in the generated code. To make an individual parameter tunable, use a MATLAB variable with a storage class other than Auto to store the parameter in memory.

You can permanently store parameter objects and other external data in a data dictionary.

This procedure uses the square-wave transfer function model `sldrtext_inlined`. To open this model, in the MATLAB Command Window, type:

```
open_system(docpath(fullfile(docroot, 'toolbox', ...
    'sldrt', 'examples', 'sldrtext_inlined')))
```

First, install the Simulink Desktop Real-Time kernel and cd to a working folder.

- 1 Open `sldrtext_inlined` and the Scope block.

```
model = 'sldrtext_inlined';
open_system(fullfile(docroot, 'toolbox', 'sldrt', 'examples', model));
scname = [model '/Scope'];
open_system(scname)
```

- 2 In the base workspace, create a parameter object configured to store the parameter as a global variable.

```
Dmp = Simulink.Parameter([1 70 10000]);
Dmp.StorageClass='ExportedGlobal';
```

- 3 Replace the Transfer Fcn block parameter Denominator with the parameter object.

```
xfername = [model '/Transfer Fcn'];
set_param(xfername, 'Denominator', 'Dmp');
```

- 4 Start execution with the original Dmp variable value.

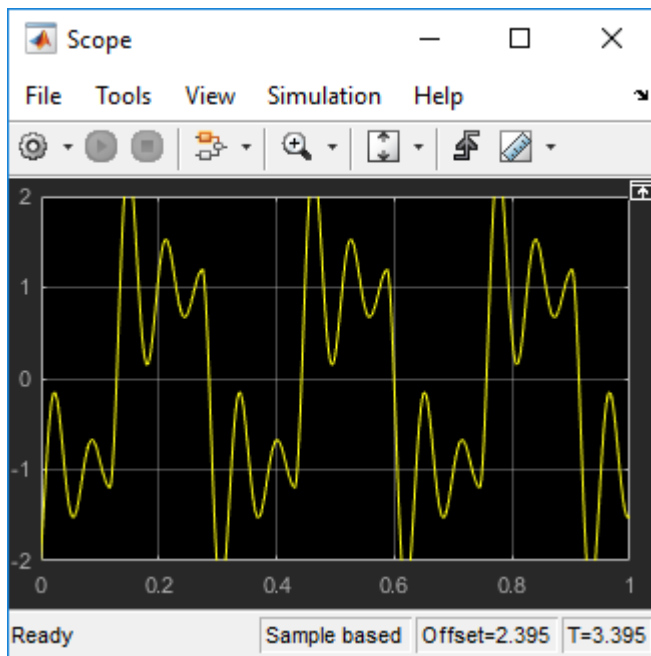
```
set_param(model, 'StopTime', 'Inf');
set_param(model, 'SimulationMode', 'external')
set_param(model, 'SimulationCommand', 'connect')
set_param(model, 'SimulationCommand', 'start')
```

- 5 Sweep the Dmp variable from 30 to 180 by 30.

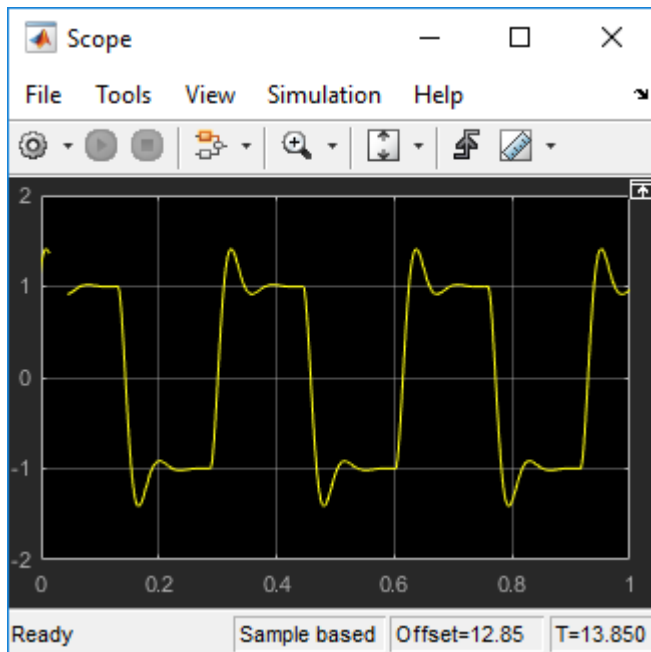
```
for Val = 30 : 30 : 180
    Dmp.Value = [1 Val 10000];
    set_param(model, 'SimulationCommand', 'update')

    pause(2.0)
end
```

The Scope block shows changes at 30-unit intervals. The figures show key changes.



Val == 30



Val == 90



`Val == 180`

6 Stop execution.

```
set_param(model, 'SimulationCommand', 'stop');
```

See Also

More About

- “Store Data in Dictionary Programmatically”
- “Tunable Block Parameters and Tunable Global Parameters” on page 3-51

Boards, Blocks, and Drivers

Simulink Desktop Real-Time software includes driver blocks for more than 200 I/O boards. These driver blocks connect the physical world to your real-time application.

- “Use I/O Boards” on page 4-2
- “Use I/O Driver Blocks” on page 4-5
- “Use Analog I/O Drivers” on page 4-9
- “Use Vector CAN Drivers” on page 4-12
- “Use Video Input Driver (Mac OS)” on page 4-13

Use I/O Boards

Typically I/O boards are preset from the factory for certain base addresses, voltage levels, and unipolar or bipolar modes of operation. Boards often include switches or jumpers that allow you to change many of these initial settings. For information about setting up and installing an I/O board, read the board manufacturer documentation.

For an online list of I/O boards that Simulink Desktop Real-Time software supports, see www.mathworks.com/hardware-support/simulink-desktop-real-time.html.

Install and Configure I/O Boards and Drivers

A Simulink Desktop Real-Time model connects to a board by including an I/O driver block. This block provides an interface to the device driver of the board and the board-specific settings. The device drivers included with the Simulink Desktop Real-Time software usually provide the same flexibility of settings offered by the board manufacturer. You can enter I/O board settings by using the I/O Block Parameters dialog box; setting jumpers and switches on the board; or both. The three types of board settings are:

- **Software selectable** — Specify the desired settings in the I/O Block Parameters dialog box. The driver writes the settings you specify to the board. Examples include A/D gain inputs and selecting unipolar or bipolar D/A outputs.
- **Hardware selectable, software readable** — Specify the desired settings by configuring jumpers or switches on the board. The driver reads the settings you selected and displays them in the I/O Block Parameters dialog box.
- **Hardware selectable, not software readable** — Set jumpers or switches on the physical board. Enter the same settings in the I/O Block Parameters dialog box. These entries must match the hardware jumpers or switches you set on the board. Use this type of setting when the board manufacturer does not provide a means for the I/O driver to write or read the board settings. Examples include base address, D/A gain, and differential or single-ended A/D inputs.

You can configure a Simulink Desktop Real-Time model to use an I/O board whether the board exists in the computer. However, you cannot run the model until the board is installed with its jumpers and switches set. Details of installation and configuration depend on the data transfer direction and the specific board, but are similar. Details for various types of boards and drivers appear later in this topic.

The following instructions configure the HUMUSOFT® AD512 I/O board for analog input. They assume that you physically configure and install the board in your computer before you add its driver to your model. To achieve the results you need, customize the steps.

To install and configure an I/O board and its driver,

- 1 Install the board in the computer, setting jumpers or switches according to the board documentation.
- 2 Open the Simulink Library Browser. In the Simulink Editor, on the **Simulation** tab, click **Library Browser**.
- 3 Drag an Analog Input I/O driver block into your model from the Simulink Desktop Real-Time library.
- 4 Double-click the driver block in the model.

- 5 In the I/O Block Parameters dialog box, click **Install new board**. From the list that appears, point to a manufacturer, then select a board type. For example, point to **Humusoft**, then click **AD512**.

The I/O board dialog box opens. The name of this dialog box depends on which I/O board you selected.

- 6 If using a PCI bus board, enter the logical device number in the **Device order** box or check **Auto-detect**.
- 7 Set the other required block parameters using the I/O Block Parameters dialog box.
- 8 Click **Test**.

The Simulink Desktop Real-Time kernel tries to connect to the selected board, and if it does so without an error, displays a message indicating that it found the board.

- 9 Click **OK** in the message box, and again in the I/O Block Parameters dialog box.

The I/O Block Parameters dialog box closes, and the parameter values are included in your Simulink model.

Multiple Boards of Identical Type

When multiple boards of identical type exist, execute the complete installation sequence for each board as if the boards were of different types. Thus, two identical PCI boards result in two entries in the list of installed boards. The entries differ only in the logical device number shown in the **Device order** box for each board.

Autodetect Multiple Boards

The Autodetect feature cannot be used to locate multiple boards of the same type. Specify their logical device numbers manually.

Multiple Driver Blocks for One Board

When you have used the I/O Block Parameters dialog box to configure a board driver, you can add additional I/O driver blocks for the same board. Drag each driver block into the model, open its I/O Block Parameters dialog box, and select the board from the list of installed boards.

Scope of Driver Block Parameters

I/O driver blocks that use a given board share identical parameters. You specify these parameters only once, when you first add the board and configure its driver. If you change a parameter in the driver block for a board, the same change occurs in the other driver blocks connected to that board.

PCI Bus Board

You do not have to set a base address for a PCI board. The plug-and-play feature of the operating system assigns a PCI slot number and logical device number. You can enter the logical device number in the **Device order** box, or you can let the driver determine the device number for you. The **Device order** box is in the I/O board dialog box, which you can open from an I/O driver Block Parameters dialog box.

Before you use a PCI or PCMCIA board, install the drivers supplied by the board manufacturer. The Simulink Desktop Real-Time software does not use these manufacturer-supplied drivers. However, they sometimes initiate the plug-and-play recognition of the board. Without these drivers installed, some boards are invisible to your computer and to the Simulink Desktop Real-Time software.

Write PCI Bus Board Drivers

Simulink Desktop Real-Time applications cannot use DLLs and kernel-mode drivers, which are not suitable for real-time operation. The device drivers supported by the Simulink Desktop Real-Time software are listed at www.mathworks.com/hardware-support/simulink-desktop-real-time.html. If no driver is listed for the board that you want to use, you can sometimes write a custom device driver.

A user-written custom device driver must program the board directly at the register level. All supported Simulink Desktop Real-Time drivers use this technique. The Simulink Desktop Real-Time software supports I/O mapped board registers for custom device drivers. The Simulink Desktop Real-Time software does not support memory-mapped board registers for custom device drivers.

To report that you require support for an unsupported board, contact MathWorks Technical Support at www.mathworks.com/contact_TS.html.

Compact PCI Board

If you use a compact PCI board (PXI®, PXI Express) use a compact PC (industrial PC). Also, install the operating system, the MATLAB environment, Simulink software, and Simulink Desktop Real-Time software on the compact PC.

PCMCIA Board

The plug-and-play feature of the operating system assigns a base address automatically. You can enter this address in the I/O board dialog box, or you can let the driver determine the address for you. You open the I/O board dialog box from an I/O driver Block Parameters dialog box.

Before you use a PCI or PCMCIA board, install the drivers supplied by the board manufacturer. Simulink Desktop Real-Time software does not use these manufacturer-supplied drivers. However, they sometimes initiate the plug-and-play recognition of the board. Without these drivers installed, some boards are invisible to your computer and to the Simulink Desktop Real-Time software.

See Also

More About

- “Run Confidence Test” on page 2-6

External Websites

- www.mathworks.com/hardware-support/simulink-desktop-real-time.html

Use I/O Driver Blocks

Simulink Desktop Real-Time I/O driver blocks allow you to select and connect specific analog channels and digital lines to your Simulink model through I/O driver blocks. These blocks provide an interface to your physical I/O boards and your real-time application. They enable the C code created by Simulink Coder code generation software to map block diagram signals to the corresponding I/O channels. All I/O blocks support all applicable Simulink data types.

You can have multiple I/O blocks associated with each type of I/O board. For example, you can have one Analog Input block for channels 1–4 and another block for channels 5–8. Each I/O block in a model specifies its own block configuration parameters, which apply only to that instance of that block.

The capability of an I/O driver block is available only if the corresponding I/O hardware device supports that capability. For example, data-oriented devices like Serial Port and File support packet and stream I/O blocks. However, data acquisition devices do not support the packet and stream I/O blocks.

The Simulink Desktop Real-Time Library provides blocks that you can use with supported I/O boards. You can also create your own I/O blocks to work with Simulink Desktop Real-Time software. See “Custom I/O Driver Basics” on page A-2 for details.

View Simulink Desktop Real-Time Library

I/O driver blocks are available in the Simulink Desktop Real-Time Library. To view this library from the MATLAB Command Window, type:

```
sldrtlib
```

To view the Simulink Desktop Real-Time Library from a model:

- 1 In the Simulink Editor, on the **Simulation** tab, click **Library Browser**.

The Simulink Library Browser opens. The left pane shows a hierarchy of libraries and categories, with the Simulink library at the top. The right pane shows the blocks available in the category selected on the left.

- 2 In the left column, double-click **Simulink Desktop Real-Time**.

The Simulink Desktop Real-Time library opens.

You can add an I/O block in the library to your Simulink model by dragging it from the library to the model. After you add the block, connect it to your model as you would any other block, and provide block configuration parameter values.

Route Signals from an I/O Block

I/O driver blocks output multiple signals as a vector instead of individual channels or lines. To connect the individual channels and lines to parts of your Simulink model, separate the vector with a Demux block.

After you add and configure an I/O driver block in your Simulink model, you can separate and connect the output signals from the blocks:

- 1 In the Simulink Editor, on the **Simulation** tab, click **Library Browser**.
- 2 In the **Simulink** library, click **Signal Routing**. From the list in the right pane, click and drag Demux to your Simulink model.
- 3 Double-click the Demux block. The Block Parameters: Demux dialog box opens. Enter the number of lines leaving the Demux block. For example, if you entered three channels in the Analog Input block, enter 3 in the **Number of outputs** box.
- 4 Click **OK**.
- 5 Connect the Analog Input block to the Demux block input.
- 6 Connect each of the Demux block output lines to the input of other blocks.
- 7 On the **Debug** tab, click **Diagnostics > Information Overlays > Nonscalar Lines**.
- 8 On the **Debug** tab, click **Diagnostics > Information Overlays > Signal Dimensions**.

Note In this example, inputs 1 and 2 are not connected, but they could be connected to other Simulink blocks.

Configure Channel Selection

To show how to specify device settings when using both analog and digital signals, this example uses the Keithley® Metrabyte DAS-1601 I/O board. The following is a specification summary of the DAS-1601 board:

- **Analog input (A/D)** — 16 single-ended or 8 differential analog inputs (12-bit), polarity is switch configured as either unipolar (0-10 volts) or bipolar (± 10 volts). Gain is software configured to 1, 10, 100, and 500.
- **Digital input** — Four unidirectional digital inputs
- **Analog output (D/A)** — Two analog outputs (12-bit). Gain is switch configured as 0-5 volts, 0-10 volts, ± 5 volts, or ± 10 volts
- **Digital output** — Four unidirectional digital outputs
- **Base address** — Switch configured base address

This section explores different configurations for input signals.

Once an Analog Input block has been placed in the model and the I/O board selected and configured, you can set up the Analog Input block to handle input signals.

Single analog input — The most basic case is a single analog input signal that is physically connected to the first analog input channel on the board. In the Block Parameter: Analog Input dialog box, and the **Input channels** box, enter:

1 or [1]

The use of brackets is optional for a single input.

Input vector with differential analog — Number the analog channels from channel 1 up to the maximum number of analog signals supported by the I/O board.

In the case of the DAS-1601, when configured as differential inputs, eight analog channels are supported. The analog input lines are numbered 1 through 8. The complete input vector is:

[1 2 3 4 5 6 7 8] or [1:8]

If you want to use the first four differential analog channels, enter

[1 2 3 4]

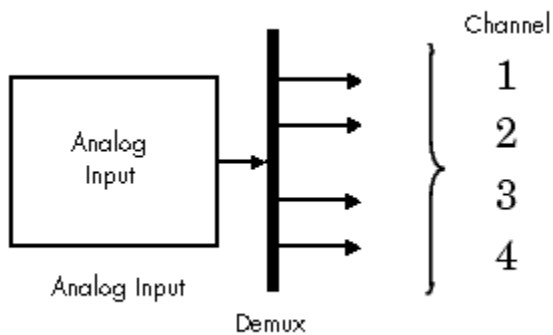
Input vector with single-ended analog — Assume that your DAS-1601 board is configured to be single-ended analog input. In this case, 16 analog input channels are supported. The complete input vector is:

[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16] or [1:16]

To use the first four single-ended analog input channels, enter:

[1 2 3 4] or [1:4]

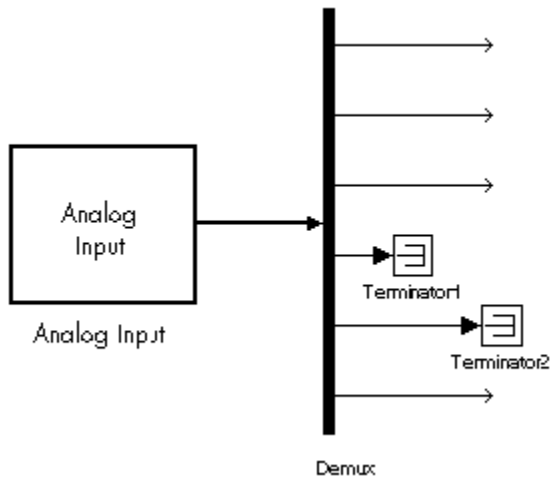
The next figure shows the resulting block diagram.



Do not specify more channels than you actually use in your block diagram. Specifying unused channels results in additional overhead for the processor with A/D or D/A conversions. In this case, for example, even though some channels are not used in the block diagram, these channels are still converted.

You could attach terminator blocks to channels 4 and 5 inside your block diagram after passing the Analog Input block vector in to a Demux block. Adding terminator blocks provides you with graphical information in your block diagram that clearly indicates which channels you connected and which are available. The penalty is that even the terminated channels are converted, adding some computational overhead.

The next figure shows the block implementation.



Depending on the board and the number of channels used, I/O conversion time can affect the maximum sample rate that can be achieved on your system. Rather than converting unused channels, specify only the set of channels that your model actually requires.

See Also

More About

- "Run Confidence Test" on page 2-6

Use Analog I/O Drivers

Control systems have unique requirements for I/O devices used with Simulink Desktop Real-Time applications. For information about writing custom I/O device drivers to work with Simulink Desktop Real-Time applications, see “Custom I/O Driver Basics” on page A-2.

Configure I/O Driver Characteristics

Simulink Desktop Real-Time applications use I/O boards provided by many hardware vendors. These boards are often used for data acquisition independently of Simulink Desktop Real-Time software. In such environments, board manufacturers usually provide their own I/O device drivers for data acquisition purposes. This use differs significantly from the behavior of drivers provided with Simulink Desktop Real-Time software.

In data acquisition applications, data is often collected in a burst or frame consisting of many points, possibly 1,000 or more. The burst of data becomes available once the final point is available. This approach is not suitable for automatic control applications, because it results in unacceptable latency for most of the data points.

In contrast, drivers used by Simulink Desktop Real-Time applications capture a single data point at each sample interval. The software gives considerable effort to minimize the latency between collecting a data point and using the data in the control system algorithm. Sometimes a board can specify a maximum sample rate (for data acquisition) higher than the rates achievable by Simulink Desktop Real-Time applications. For data acquisition, such boards usually acquire data in bursts and not in the point-by-point fashion required by control systems.

Normalize Scaling for Analog Inputs

Simulink Desktop Real-Time software allows you to normalize I/O signals internal to the block diagram. Generally, inputs represent real-world values such as angular velocity, position, temperature, and pressure. This ability to normalize signals allows you to

- Apply your own scale factors
- Work with meaningful units without having to convert from voltages

When using an Analog Input block, you select the range of the external voltages that the board receives, and you select the block output signal. For example, you could set the voltage range to 0 to +5 V, and the block output signal as `Normalized unipolar`, `Normalized bipolar`, `Volts`, or `Raw`.

If you prefer to work with units of voltage within your Simulink block diagram, you can select `Volts`.

To apply your own scaling factor, select `Normalized unipolar` or `Normalized bipolar`, add a Gain block, and add an offset to convert the value to a meaningful value in your model.

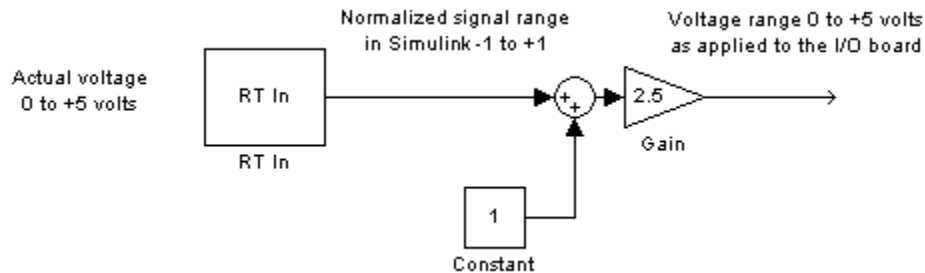
If you prefer nonrounded integer values from the analog-to-digital conversion process, you can select `Raw`.

0 to +5 Volts and Normalized bipolar

From the Input range list, select `0 to +5 V`, and from the Block output signal list, select `Normalized bipolar`. This example converts a normalized bipolar value to volts, but you could also easily convert directly to another parameter in your model.

0 to 5 volts --> $([-1 \text{ to } 1] \text{ normalized} + 1) * 2.5$

In your block diagram, you can convert the normalized value to volts as follows.

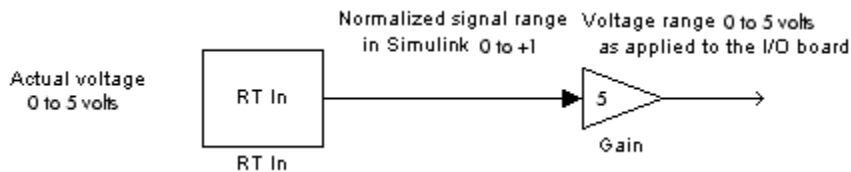


0 to +5 V and Normalized unipolar

From the Input range list, select 0 to +5 V, and from the Block output signal list, select Normalized unipolar. This example converts a normalized unipolar value to volts, but you could also easily convert directly to another parameter in your model.

0 to 5 volts --> $([0 \text{ to } 1] \text{ normalized} * 5.0)$

In your block diagram, you can convert the normalized value to volts as follows.

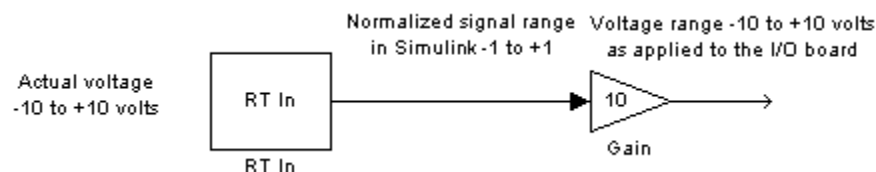


-10 to +10 V and Normalized bipolar

From the Input range list, select -10 to +10 V, and from the Block output signal list, select Normalized bipolar. This example converts a normalized bipolar value to volts, but you could also easily convert directly to another parameter in your model.

-10 to 10 volts --> $([-1 \text{ to } +1] \text{ normalized} * 10)$

In your block diagram, you can convert the normalized value to volts as follows.

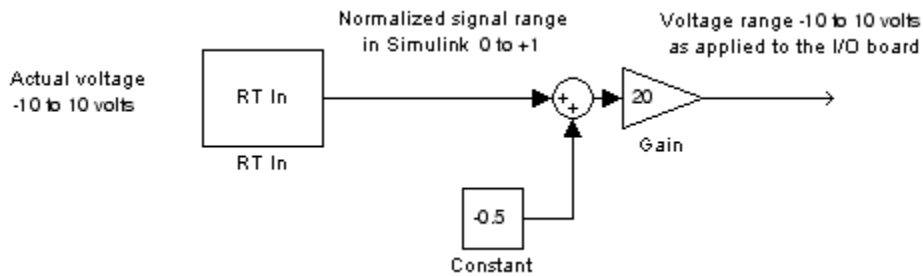


-10 to +10 V and Normalized unipolar

From the Input range list, select -10 to +10 V, and from the Block output signal list, select Normalized unipolar. This example converts a normalized bipolar value to volts, but you could also easily convert directly to another parameter in your model.

-10 to 10 volts --> $([0 \text{ to } 1] \text{ normalized} - 0.5) * 20$

In your block diagram, you can convert the normalized value to volts as follows.



Normalize Scaling for Analog Outputs

Analog outputs are treated in an equivalent manner to analog inputs.

For example, assume that the voltage range on the D/A converter is set to 0 to +5 volts and the Block input signal is selected as Normalized bipolar. With this configuration, a Simulink signal of amplitude -1 results in an output voltage of 0 volts. Similarly, a Simulink signal of amplitude +1 results in an output voltage of +5 volts.

For another example, assume that the voltage range on the D/A converter is set to -10 to +10 volts and the Block input signal is selected as Normalized bipolar. With this configuration, a Simulink signal of amplitude -1 results in an output voltage of -10 volts. Similarly, a Simulink signal of amplitude +1 results in an output voltage of +10 volts.

As required by your selected voltage range, adjust your signal amplitudes using a Gain block, Constant block, and Summer block.

See Also

Analog Input | Analog Output

More About

- “Custom I/O Driver Basics” on page A-2
- “Run Confidence Test” on page 2-6

Use Vector CAN Drivers

Before you can use the Vector Informatik CAN devices, you must install the drivers for them.

- 1 Install the Vector CAN devices. See the Vector Informatik GmbH documentation for installation instructions for hardware devices such as CANcaseXL, CANboardXL, and CANcardXL, drivers, and support libraries.
- 2 Install the Vector XL driver library for the Windows XP, Windows Vista™, or Windows 7 operating systems. If you do not have this library, download it from the Vector Informatik GmbH website:

`www.vector.com`

- 3 Install the driver file.
- 4 Copy the `vxlapi.dll` file into the Windows system `root\system32` folder.
- 5 Use the Vector software to assign physical CAN channels to an application. When specifying the name for the Simulink Desktop Real-Time application, use the name `MATLAB`.

See Also

Packet Input | Packet Output

More About

- “Run Confidence Test” on page 2-6

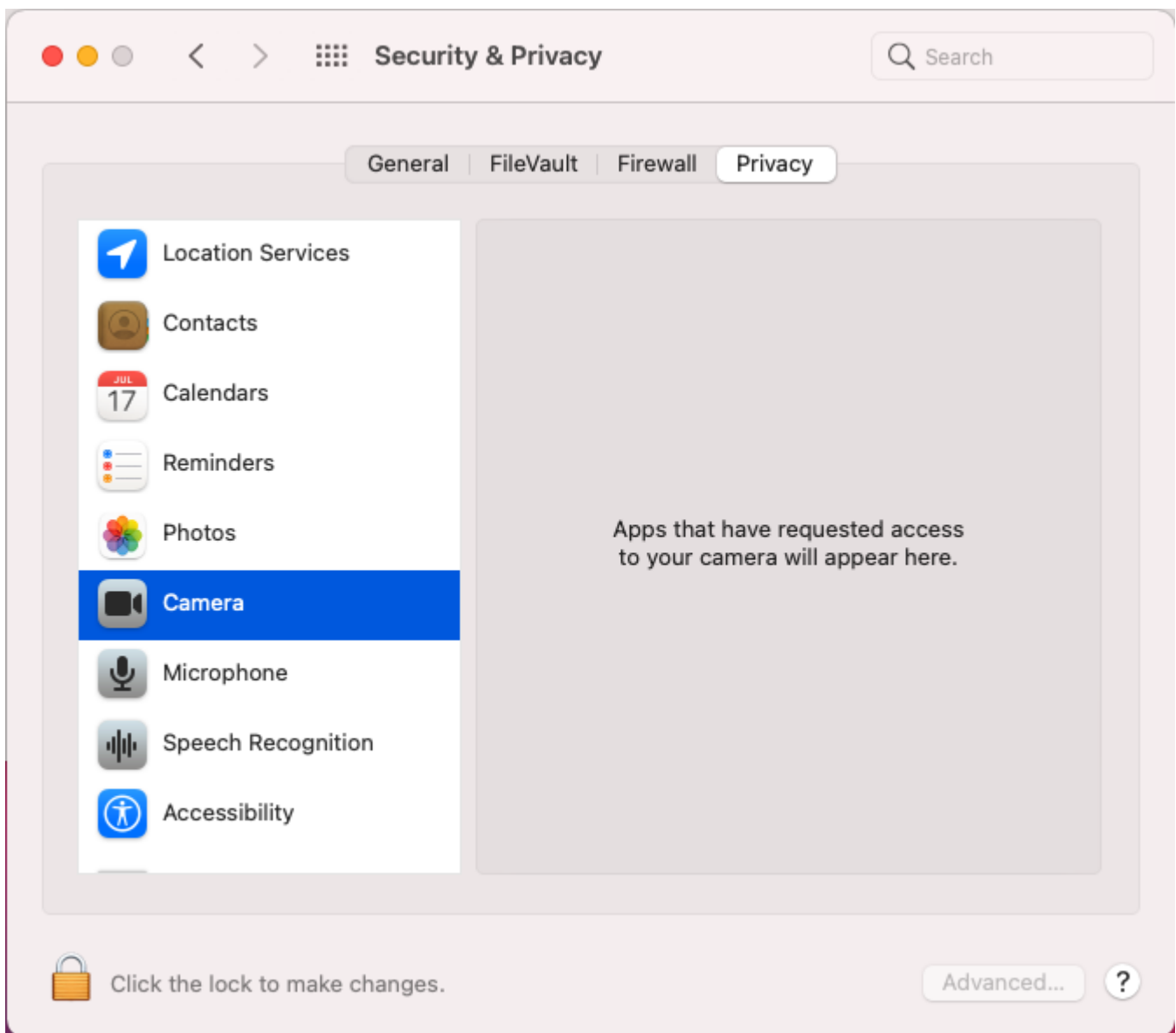
Use Video Input Driver (Mac OS)

To configure the Video Input block for Windows or Mac OS platforms, you configure the camera by using **Install new board** or **Board setup** for the Video Input block.

For model simulation on a Mac OS platform, you also configure access preferences for your Mac approved camera. If you do not set these preference, the video input does not display.

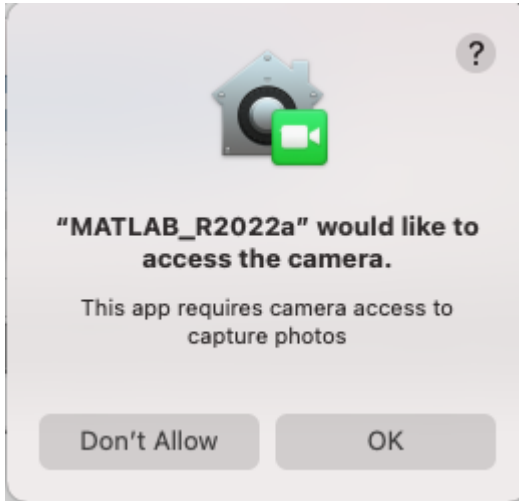
These steps describe how to check whether MATLAB has been granted access to the Mac approved camera on your system.

- 1 View the **System Preference > Security & Privacy > Camera** settings. The image shows that there are no applications that have been granted access to the camera system device.

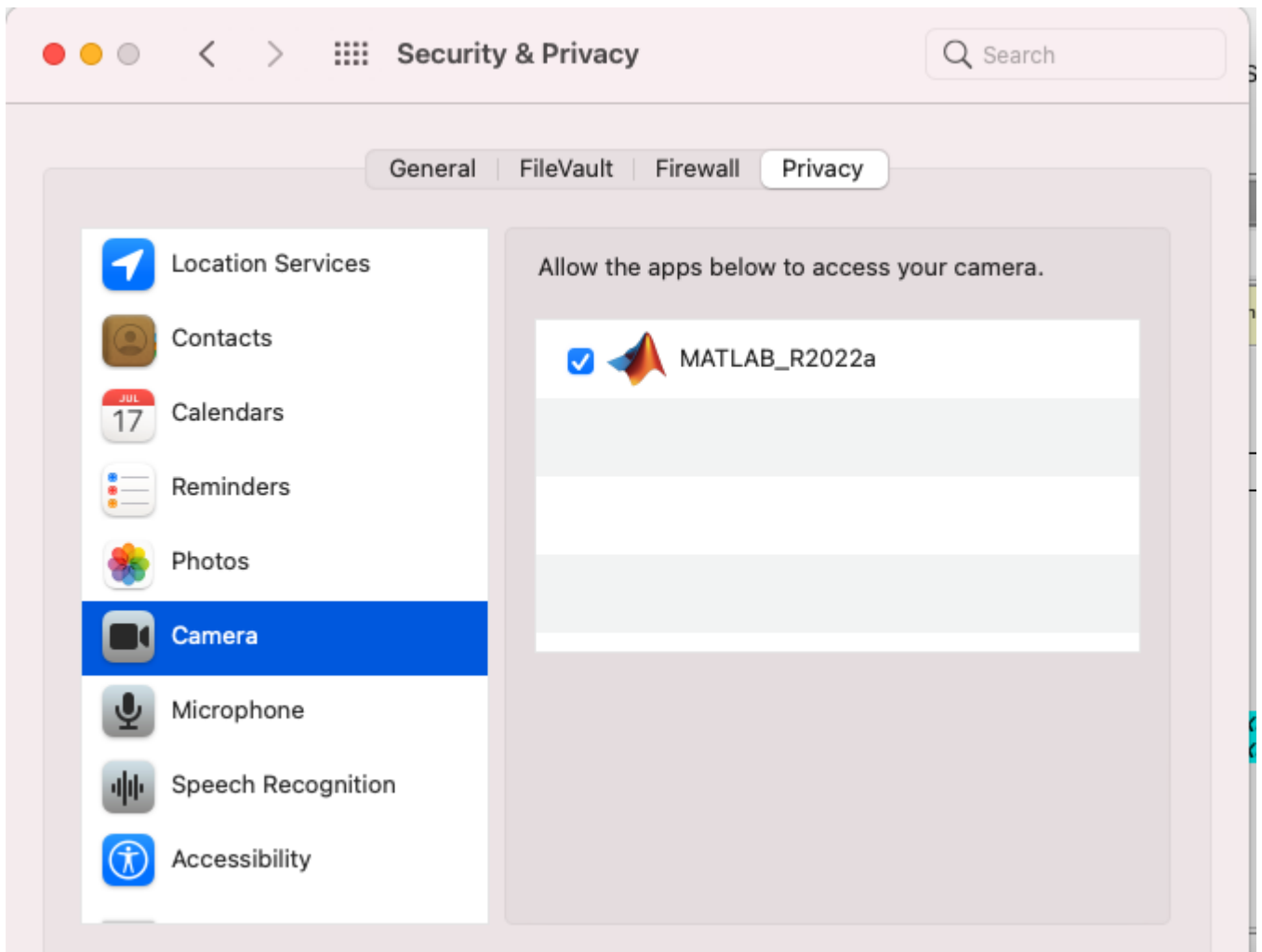


- 2 Start MATLAB from the Mac OS Finder.

- 3 When the Video Input block attempts to access the camera for the first time (for example during Connected I/O mode simulation of a model that contains a Video Input block), the system requests confirmation that this access is **OK**.



- 4 View the **System Preference > Security & Privacy > Camera** settings. After you click **OK** for the access, the MATLAB application appears in the list of applications that have been granted access to the camera system device.



If you start MATLAB at the command line in the Mac OS Terminal application, the Terminal application appears in the list instead of MATLAB.

See Also

Video Input

Related Examples

- "Video Input" on page 5-24

Custom I/O Driver Blocks

Custom I/O Driver Basics

In this section...

“Supported C Functions” on page A-2

“Unsupported C Functions” on page A-2

“Incompatibility with Operating System API Calls” on page A-3

“I/O Register Access from S-Functions Limitation” on page A-3

You can write custom I/O device drivers to work with Simulink Desktop Real-Time applications.

Note Do not use Analog Input, Analog Output, Digital Input, or Digital Output drivers as starting points for creating custom device drivers.

Supported C Functions

You can use ANSI® C functions that do not use the operating system in your custom blocks or I/O drivers. The following includes a partial list of supported functions:

- **Console I/O** — printf

The printf function sends output to the MATLAB Command Window when it is called from the real-time application.

- **Data conversion** — abs, atof, atoi, atol, itoa, labs, ltoa, strtod, strtol, strtoul, ultoa

- **Memory allocation** — calloc, free, malloc

Memory allocation is not an operation that can be done in real time. To work with a Simulink Desktop Real-Time application, memory management must occur before real-time simulation begins. Simulation switches into real time after mdlStart, so you can allocate memory in mdlInitializeSizes or mdlStart. You cannot allocate memory in any function after mdlStart, such as mdlOutputs or mdlUpdate.

- **Memory manipulation** — _memccpy, memcpy, memchr, memcmp, _memicmp, memmove, memset

- **Character string manipulation** — strcat, strchr, strcmp, strcpy, strcspn, _strdup, _stricmp, strlen, _strlwr, strncat, strncmp, strncpy, _strnset, strpbrk, strrchr, _strrev, _strset, strspn, strstr, strtok,strupr

- **Mathematical** — acos, asin, atan, atan2, ceil, cos, cosh, div, exp, fabs, floor, fmod, frexp, ldexp, ldiv, log, log10, max, min, modf, pow, rand, sin, sinh, sqrt, srand, tan, tanh, uldiv

- **Character class tests and conversion** — isalnum, isalpha, _isascii, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit, isxupper, isxlower, _toascii, tolower, toupper

- **Searching and sorting** — bsearch, qsort

- **Dummy functions** — exit, fprintf

Unsupported C Functions

If you create your own custom I/O driver blocks, use only C functions that Simulink Desktop Real-Time supports. Simulink Desktop Real-Time does not support functions that use the operating system.

This category includes functions from vendor-supplied driver libraries for the operating system, which are also not supported.

The following list includes many, but not all, of the unsupported functions:

- **File I/O** — fopen, freopen, fclose, fread, fwrite, fputs, fputc, fgets, fgetc, gets, getc, getchar, puts, putc, putchar, fflush, setbuf, setvbuf
- **Console I/O** — fprintf, sprintf, vsprintf, vprintf, vsprintf, fscanf, scanf, sscanf
- **Process management** — spawn, exit, abort, atexit
- **Signals and exceptions** — signal, longjmp, raise
- **Time functions** — clock, time, difftime, asctime, ctime, difftime, gmtime, localtime, mktime, strftime
- **Operating system API functions** — *No operating system API functions, such as Win64 functions, are supported .*

Incompatibility with Operating System API Calls

The Simulink Desktop Real-Time kernel intercepts the interrupt from the system clock. It then reprograms the system clock to operate at a higher frequency for running your real-time application. At the original clock frequency, it sends an interrupt to the operating system to allow software that uses the operating system API to run.

As a result, **software that uses the operating system API, such as Win64 functions, cannot be executed as a component of your real-time application.** Software you use to write I/O drivers must not make calls to the operating system API.

I/O Register Access from S-Functions Limitation

Operating system drivers can access I/O registers only from the real-time kernel and not from the Simulink software. To prevent drivers from attempting to access I/O registers from Simulink S-functions, enter code fragments like the following:

```
#ifndef MATLAB_MEX_FILE
/* we are in real-time kernel, do board I/O */
#else
/* we are in Simulink, don't do board I/O */
#endif
```


Simulink Desktop Real-Time Examples

Real-Time Van der Pol Simulation

This example shows a real-time version of the Simulink® Van der Pol simulation model.

This model does not need any external signals, so it does not need any data acquisition hardware or driver. The model is useful for the first time that you work with Simulink Desktop Real-Time™ because you do not have to configure I/O hardware.

Run Model in Connected IO Mode

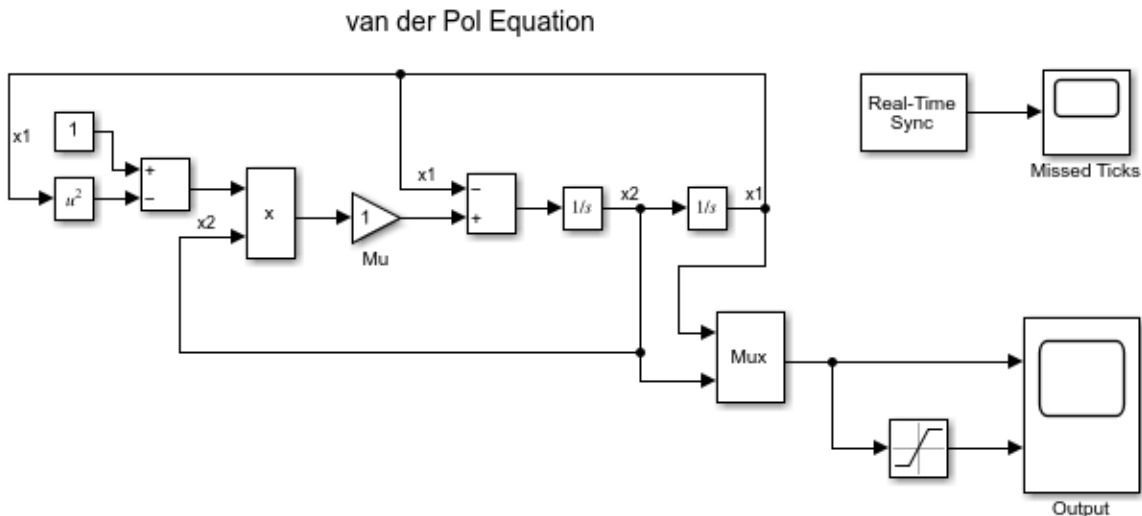
- 1 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 2 To start the real-time execution, click **Run in Real-Time**.
- 3 Observe any missed ticks on the **Missed Ticks** scope.

Run Model in Run in Kernel Mode

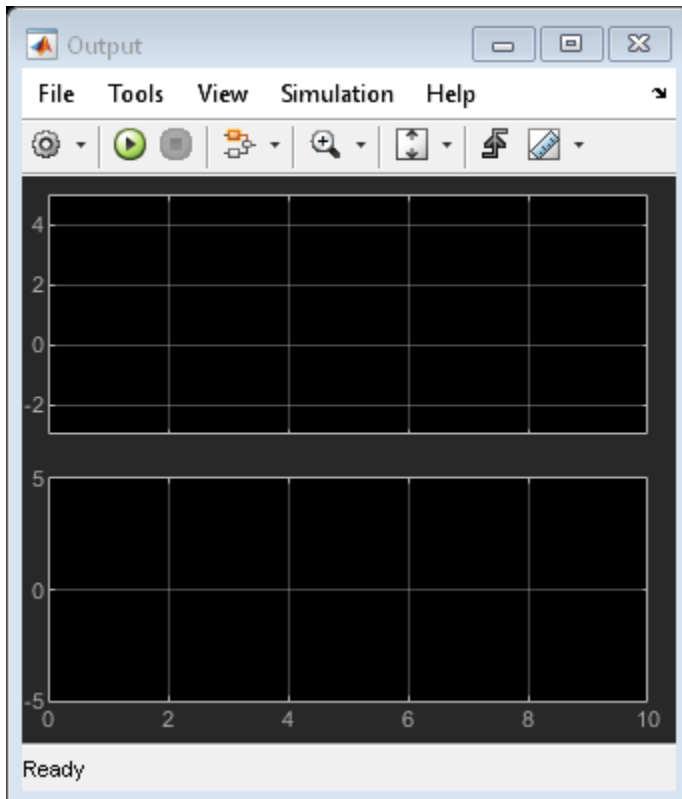
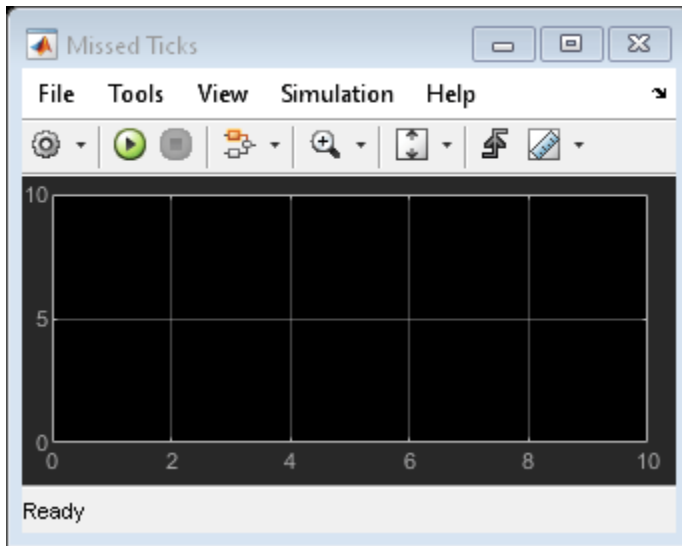
- 1 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 2 To start the real-time execution, click **Run in Real Time**. The model builds, connects to Simulink in **Run in Kernel** mode, and starts.
- 3 Observe that **Missed Ticks** is zero.

Open the Model

```
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtextamples', 'sldrtext_vdp'));
```



To run the example as synchronized simulation with I/O access, select **Connected IO mode.**
To run the example as compiled real-time executable, select **Run in Kernel mode.**



Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up the Model

```
clear
close all
bdclose all
```

See Also

- “Create a Real-Time Application” on page 3-4
- “Prepare for Real-Time Execution” on page 3-2

Water Tank Model with Dashboard

This example shows a real-time model of a water tank controlled by dashboard controls. You can change the inputs to the plant by using the dashboard knobs and observe the response on the gauges.

Run Model in Connected IO Mode

- 1 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 2 To start the real-time execution, click **Run in Real-Time**.
- 3 Change the input flow and output valve values by using the dashboard controls and observe the results on the dashboard gauges.

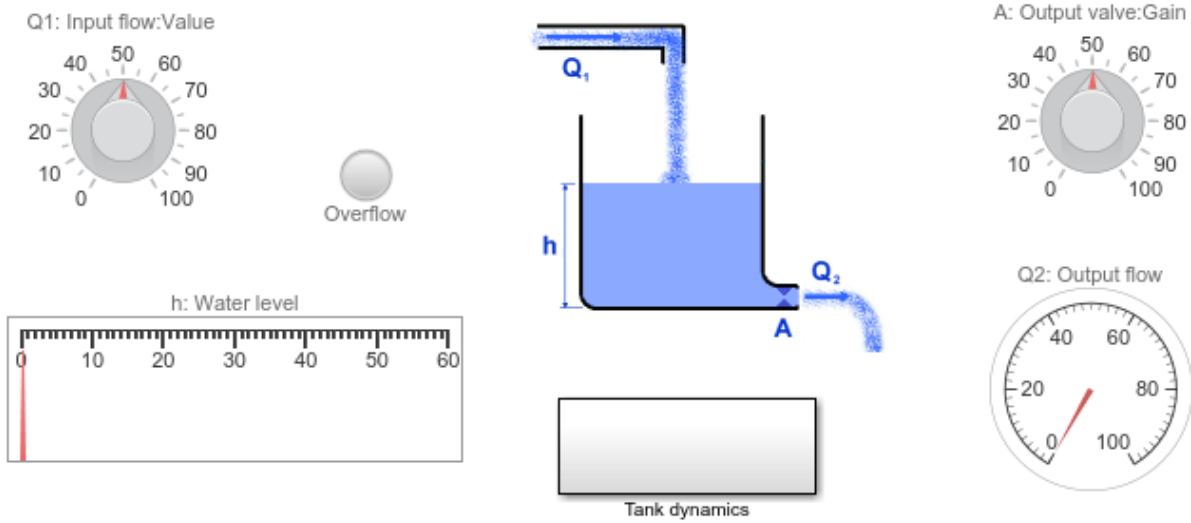
Run Model in Run in Kernel Mode

- 1 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 2 To start the real-time execution, click **Run in Real-Time**. The model builds, connects to Simulink in **Run in Kernel** mode, and starts.
- 3 Change the input flow and output valve values by using the dashboard controls and observe the results on the dashboard gauges.

Open the Model

```
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_dashboard'));
```

Water Tank Model with Dashboard

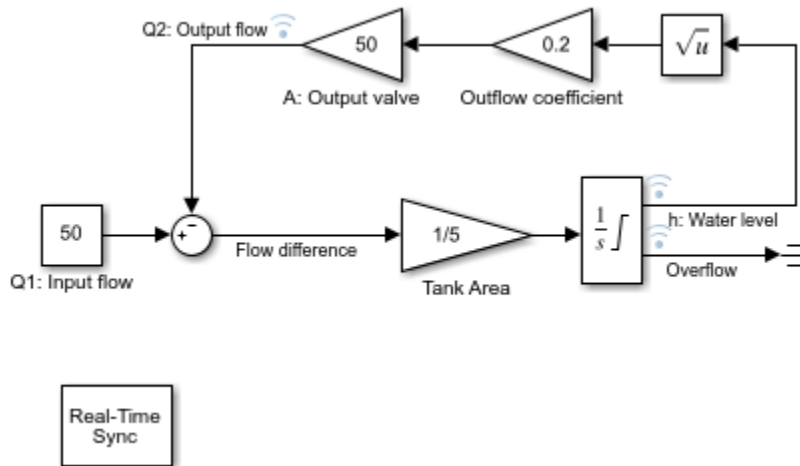


To run the example as synchronized simulation with I/O access, select **Connected IO mode**. To run the example as compiled real-time executable, select **Run in Kernel mode**.

Copyright 1994-2021 The MathWorks, Inc.

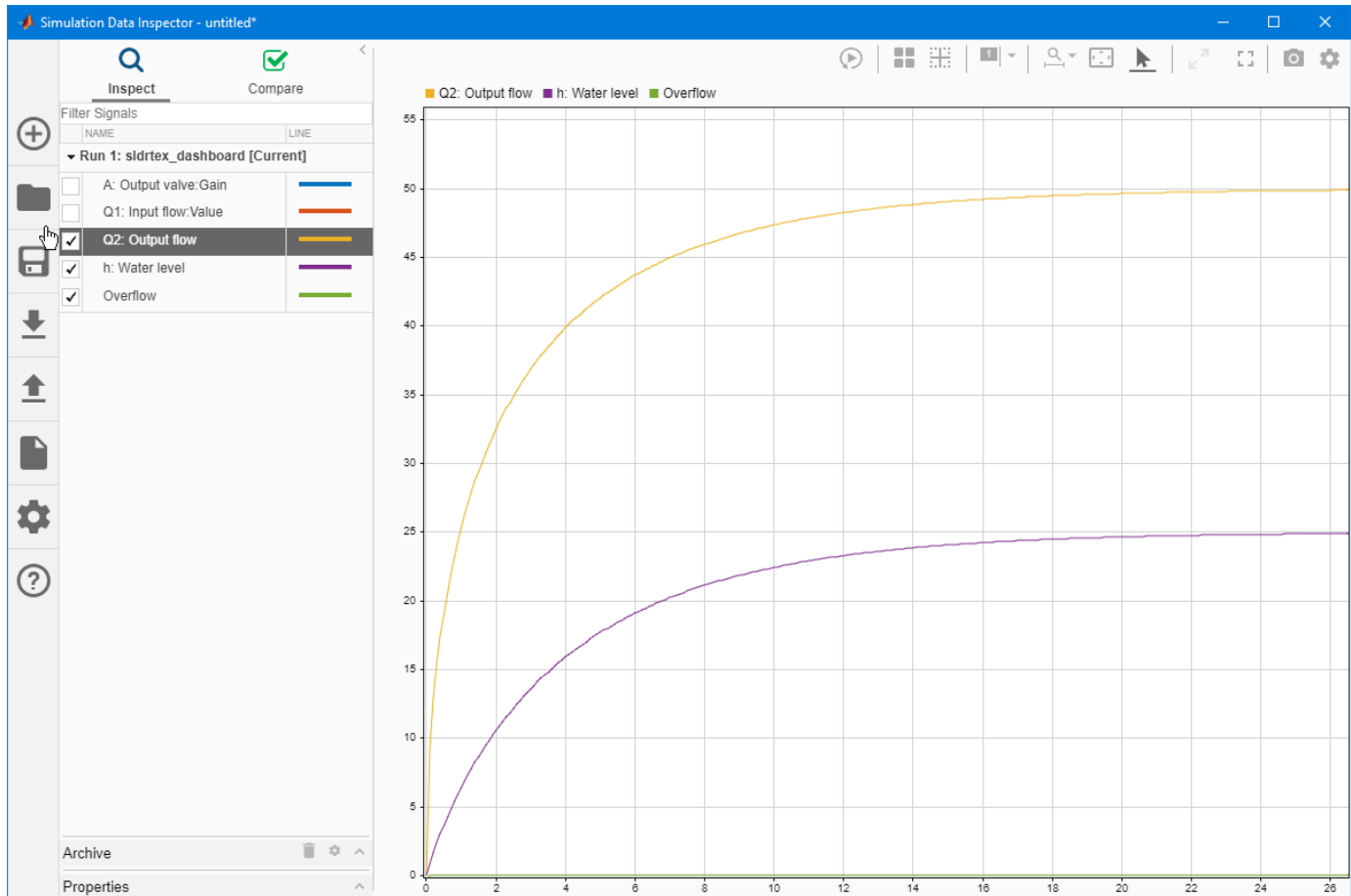
Open Subsystem and See Signals Marked for Signal Logging

- 1 Double click the Tank dynamics subsystem.
- 2 Observe the Overflow, Water level, and Output flow signals are marked for signal logging with the Simulation Data Inspector.



Run the Model and View Logged Signals

- 1 Click the **Run in Real Time** button.
- 2 Click the **Data Inspector** button.
- 3 Observe signal logging in the Simulation Data Inspector.



Clean Up the Model

```
clear
close all
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “Inspect Simulink® Desktop Real-Time™ Signals with Simulation Data Inspector” on page 3-32

Real-Time Signal Generator

This example shows how to produce an analog output signal by using Simulink Desktop Real-Time™. Because analog output can require less configuration and is easier to connect than analog input, this model is useful for working with data acquisition boards. You can verify the presence of the generated signal, for example by connecting an oscilloscope to the analog output pins of your data acquisition board.

Note: To run this model, you must have a data acquisition board connected to your computer.

Run Model in Connected IO Mode

- 1 Open the Analog Output block and select your data acquisition board. If there is no board installed, install it by clicking the **Install new board** button.
- 2 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 3 To start the real-time execution, click **Run in Real-Time**.

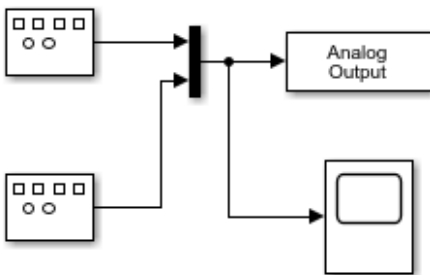
Run Model in Run in Kernel Mode

- 1 Open the Analog Output block and select your data acquisition board. If there's no board installed, install it by clicking the **Install new board** button.
- 2 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 3 To start the real-time execution, click **Run in Real Time**.

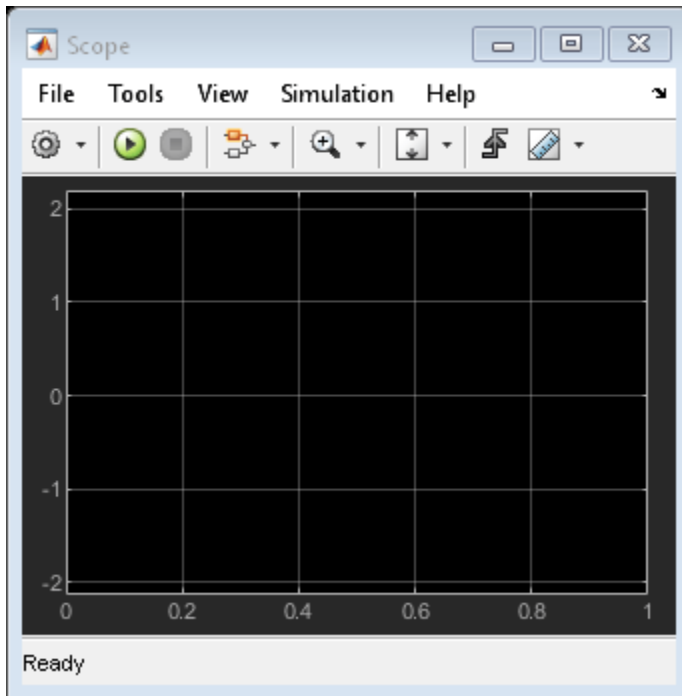
The model builds, connects to Simulink in **Run in Kernel** mode, and starts.

Open the Model

```
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_siggen'));
```



To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.



Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up the Model

```
clear  
close all  
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “Real-Time Controller” on page 5-10

Real-Time Controller

This example shows how to build a simple closed-loop real-time controller by using Simulink Desktop Real-Time™. The output of the controlled plant is connected to the analog input of your data acquisition board. This signal is subtracted from the set point value generated by the signal generator and processed by a PID controller. The output of the controller drives the input of the plant by using the analog output of your data acquisition board.

This model is a simplified version of the controller used for the <http://www.humusoft.cz/produkty/models/ce152> **CE152 Magnetic Levitation Model**.

Note: To run this model, you must have a data acquisition board connected to your computer.

Run Model in Connected IO Mode

- 1 Open the Analog Input and Analog Output blocks and select your data acquisition board. If there is no board installed, install it by clicking the **Install new board** button.
- 2 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 3 To start the real-time execution, click **Run in Real-Time**.

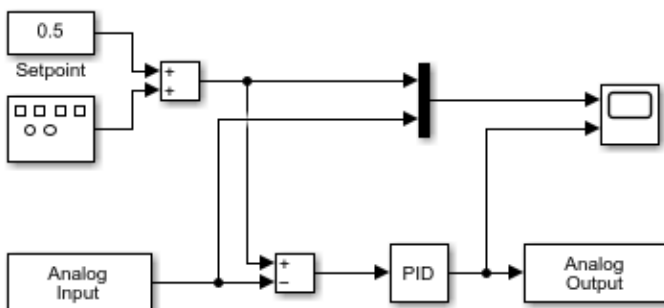
Run Model in Run in Kernel Mode

- 1 Open the Analog Input and Analog Output blocks and select your data acquisition board. If there is no board installed, install it by clicking the **Install new board** button.
- 2 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 3 To start the real-time execution, click **Run in Real Time**.

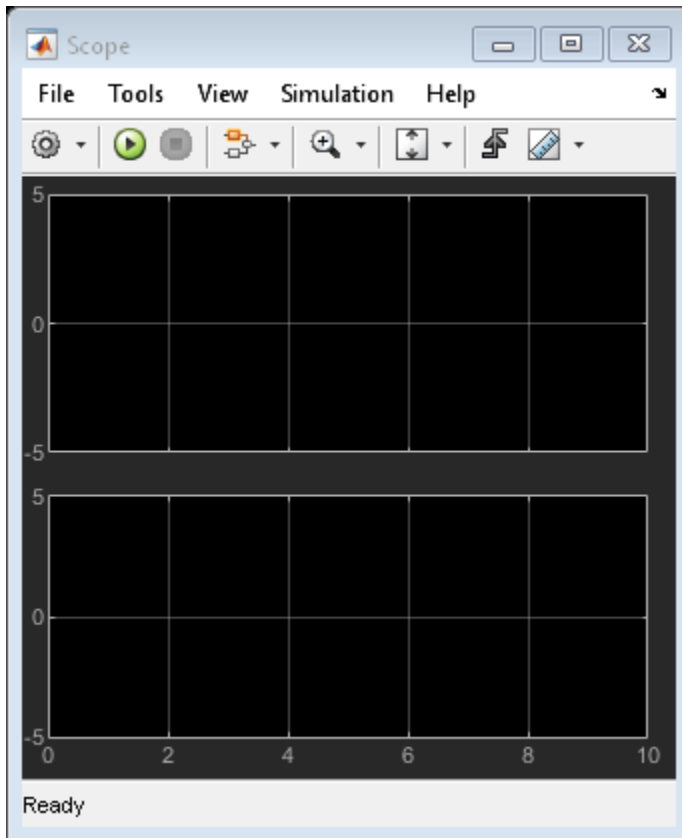
The model builds, connects to Simulink in **Run in Kernel** mode, and starts.

Open the Model

```
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_controller'));
```



To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.



Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up Model

```
clear  
close all  
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “Real-Time Filter” on page 5-12

Real-Time Filter

This example shows a real-time filter built using DSP System Toolbox™ and Simulink Desktop Real-Time™. The unfiltered signal is acquired by the analog input, passed through a filter designed by DSP System Toolbox and finally sent to analog output. Both the unfiltered and filtered signals are shown in real-time using the Scope block.

Note: To run this model, you must have a data acquisition board connected to your computer. This model requires DSP System Toolbox.

Run Model in Connected IO Mode

- 1 Open the Analog Input and Analog Output blocks and select your data acquisition board. If there is no board installed, install it by clicking the **Install new board** button.
- 2 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 3 To start the real-time execution, click **Run in Real-Time**.

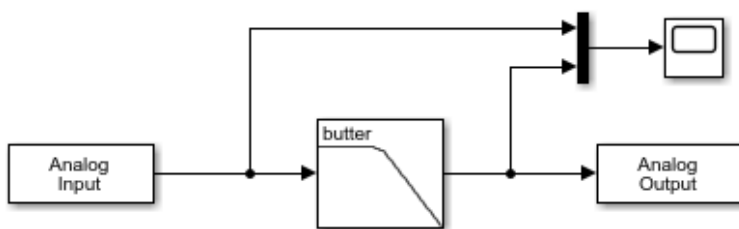
Run Model in Run in Kernel Mode

- 1 Open the Analog Input and Analog Output blocks and select your data acquisition board. If there is no board installed, install it by clicking the **Install new board** button.
- 2 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 3 To start the real-time execution, click **Run in Real Time**.

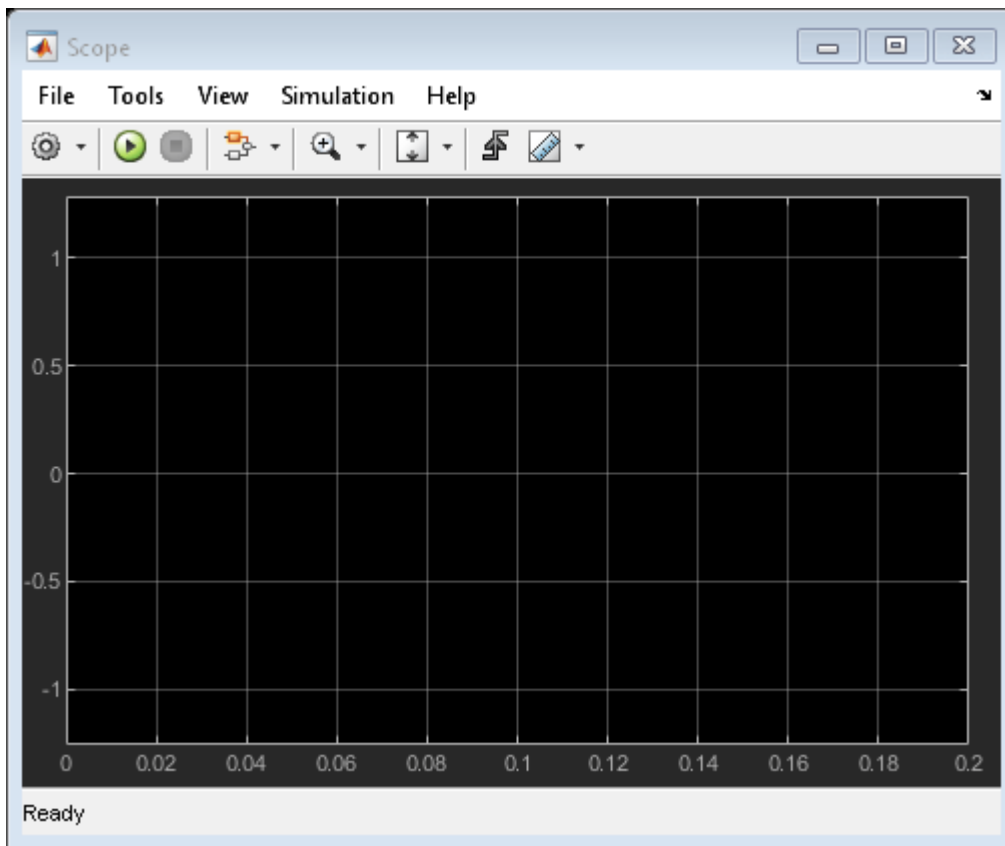
The model builds, connects to Simulink in **Run in Kernel** mode, and starts.

Open the Model

```
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_filter'));
```



To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.



Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up Model

```
clear  
close all  
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “Real-Time Controller” on page 5-10

Frequency Measurement

This example shows how to measure input signal frequency by using Simulink Desktop Real-Time™. The measured signal is connected to the counter input of your data acquisition board. Counter Input block is configured to reset counter each sample-hit after counter read. Dividing counter value by sample time gives input signal frequency.

Note: To run this model, you must have a data acquisition board with counter input connected to your computer.

Run Model in Connected IO Mode

- 1 Open the Counter Input block and select your data acquisition board. If there is no board installed, install it by clicking the **Install new board** button.
- 2 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 3 To start the real-time execution, click **Run in Real-Time**.

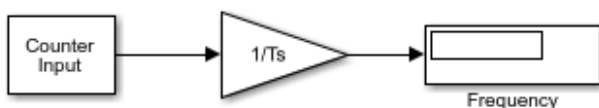
Run Model in Run in Kernel Mode

- 1 Open the Counter Input block and select your data acquisition board. If there is no board installed, install it by clicking the **Install new board** button.
- 2 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 3 To start the real-time execution, click **Run in Real-Time**.

The model builds, connects to **Run in Kernel** mode, and starts.

Open the Model

```
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtextamples', 'sldrtext_counter'));
```



To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.

Copyright 1994-2021 The MathWorks, Inc.

Close Open Scopes

```
close_system(find_system(gcs, 'BlockType', 'Scope'));
```

Clean Up Model

```
clear
close all
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “PWM Frequency and Duty Measurement” on page 5-16

PWM Frequency and Duty Measurement

This example shows how to measure PWM signal frequency and duty using Simulink Desktop Real-Time™. The measured signal is connected to gate pins of two counter inputs of your data acquisition board. The first Counter Input block is configured to measure signal duty by measuring the time between rising and falling edge of the signal. The other Counter Input block is configured to measure signal period by measuring the time between two rising edges of the signal. Both the counters use internal on-board clock as the clock source. PWM frequency and duty are then computed based on these values.

Note: To run this model, you must have a supported data acquisition board connected to your computer.

Run Model in Connected IO Mode

This model is pre-set to be used with the National Instruments PCIe-6323 board. You can use it with any board from the PCIe-63xx series. To change the board, open both the Counter Input blocks and select your data acquisition board. You may need to register the board by clicking the **Install new board** button first.

Alternatively, you can also use any board from the PCI-62xx series or the PCI-60xx series. For these boards, please change the Counter base frequency from 100 MHz to 80 MHz or 20 MHz, respectively.

- 1 Connect your PWM signal to both CTR 0 GATE and CTR 1 GATE input pins. To measure both PWM frequency and duty, two counter channels are required.
- 2 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 3 To start the real-time execution, click **Run in Real-Time**.

Run Model in Run in Kernel Mode

This model is pre-set to be used with the National Instruments PCIe-6323 board. You can use it with any board from the PCIe-63xx series. To change the board, open both the Counter Input blocks and select your data acquisition board. You may need to register the board by clicking the **Install new board** button first.

Alternatively, you can also use any board from the PCI-62xx series or the PCI-60xx series. For these boards, please change the Counter base frequency from 100 MHz to 80 MHz or 20 MHz, respectively.

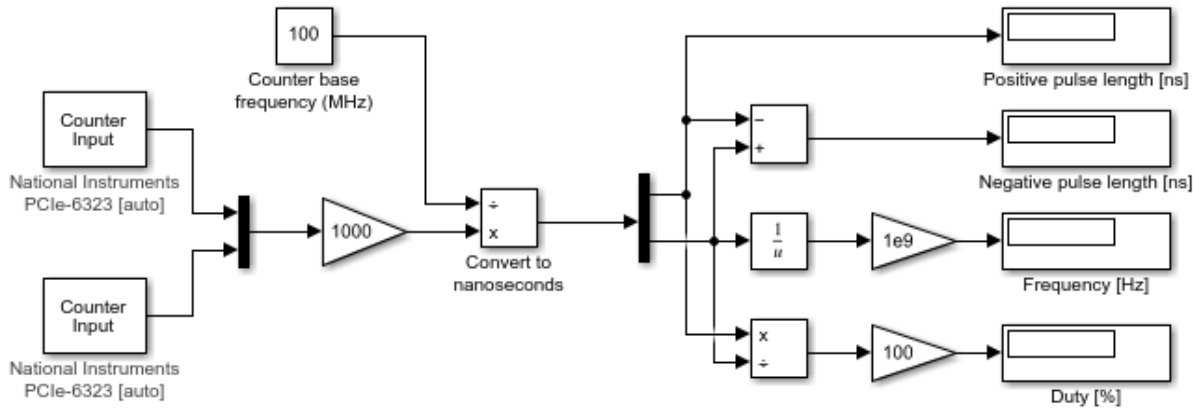
- 1 Connect your PWM signal to both CTR 0 GATE and CTR 1 GATE input pins. To measure both PWM frequency and duty, two counter channels are required.
- 2 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 3 To start the real-time execution, click **Run in Real Time**.

The model builds, connects to **Run in Kernel** mode, and starts.

Open the Model

```
warning('off','sldrt:blkgui:boardnotonlist');  
open_system(fullfile(matlabroot,'toolbox','sldrt','sldrtexamples','sldrtex_pwmmeasure'));
```


PWM Frequency and Duty Measurement



To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.

Copyright 1994-2021 The MathWorks, Inc.

Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up Model

```
clear
close all
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “Frequency Measurement” on page 5-14

Packet Input/Output

This example shows how to transfer data through UDP communication protocol using binary encoding. The model sends data within one computer, from one UDP port to another. You can modify the model to communicate between two computers by splitting this model into its send and receive parts and running the models on two computers. The yellow blocks are used to send the data, the blue blocks are used to receive the data. Then, please enter the host names or IP addresses of the two computers into the appropriate fields in the Board Setup dialog.

Run Model in Connected IO Mode

- 1 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 2 To start the real-time execution, click **Run in Real-Time**.

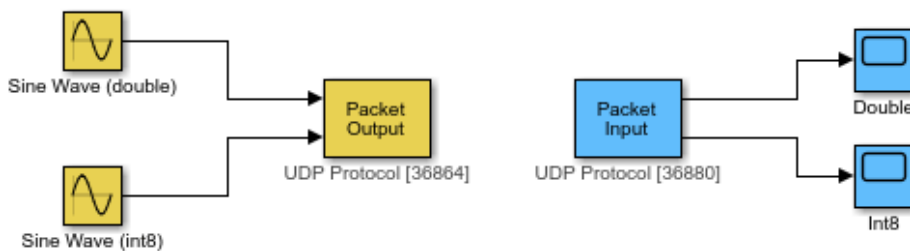
Run Model in Run in Kernel Mode

- 1 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 2 To start the real-time execution, click **Run in Real Time**. The model builds, connects to Simulink in **Run in Kernel** mode, and starts.

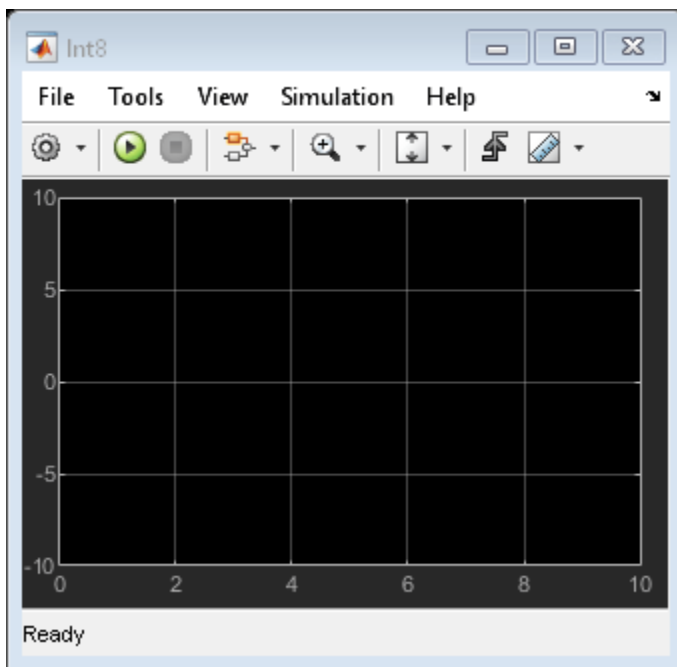
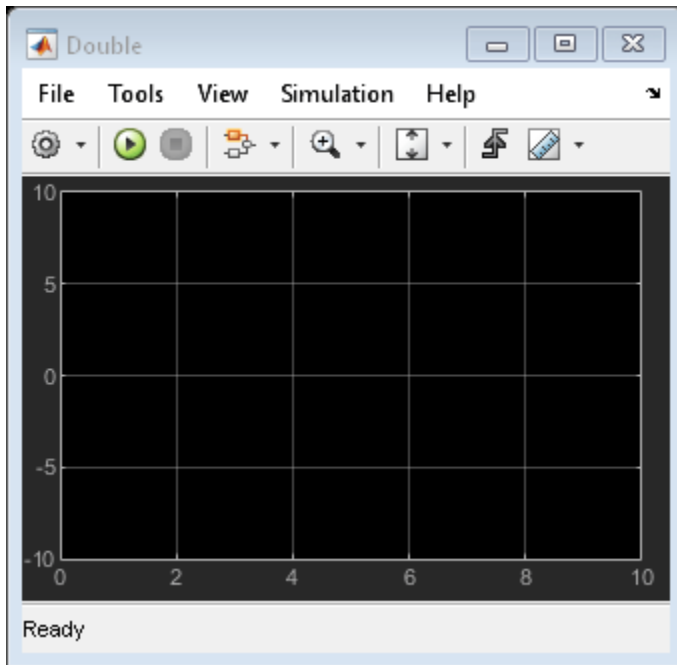
Open the Model

These commands open the model and suppress warning about board not installed.

```
w = warning('off', 'sldrt:blkgui:boardnotonlist');
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_packetio'));
warning(w);
```



To run the example as synchronized simulation with I/O access, select **Connected IO mode. To run the example as compiled real-time executable, select **Run in Kernel** mode.**



Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up Model

```
clear  
close all  
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “Stream Input/Output” on page 5-21

Stream Input/Output

This example shows how to transfer data through TCP communication protocol by using ASCII encoding. The model sends data within one computer, from one TCP port to another. You can modify the model to communicate between two computers by splitting this model into its send and receive parts and running the models on two computers. The yellow blocks are used to send the data, the blue blocks are used to receive the data. Then, please enter the host names or IP addresses of the two computers into the appropriate fields in the Board Setup dialog.

Run Model in Connected IO Mode

- 1 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 2 To start the real-time execution, click **Run in Real-Time**.

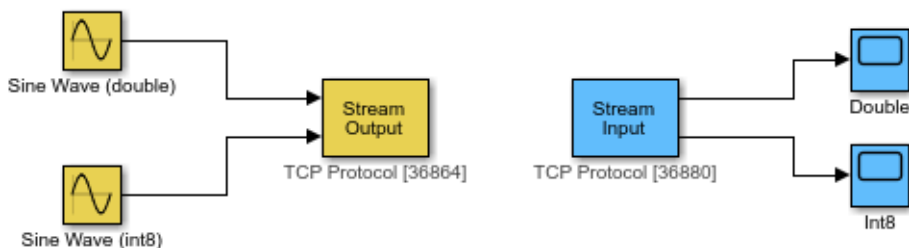
Run Model in Run in Kernel Mode

- 1 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 2 To start the real-time execution, click **Run in Real Time**. The model builds, connects to Simulink in **Run in Kernel** mode, and starts.

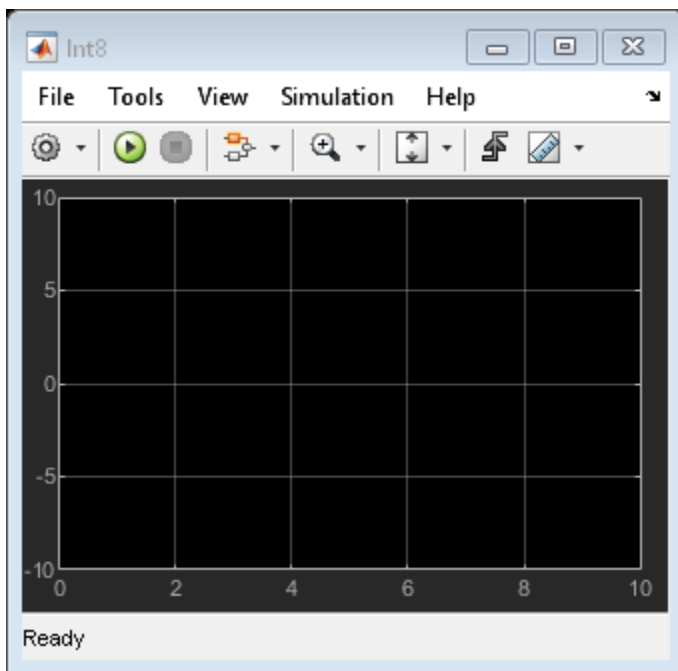
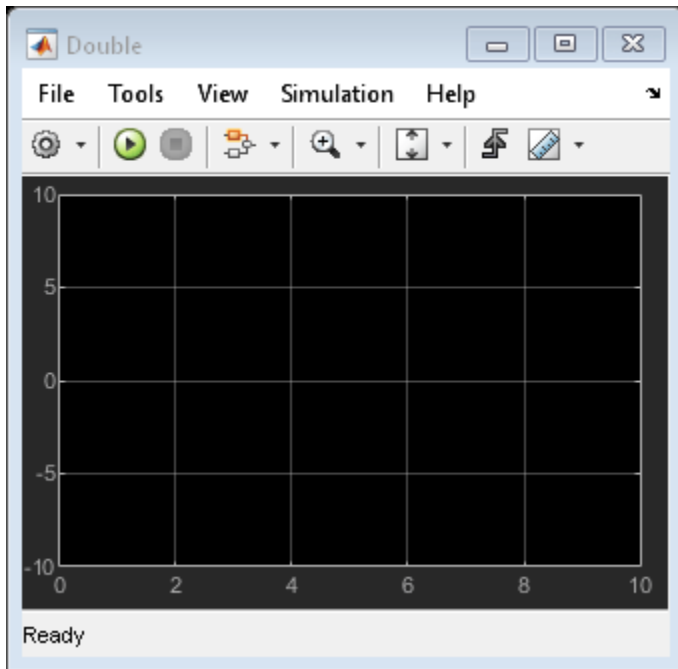
Open the Model

These commands open the model and suppress warnings about board not installed.

```
w = warning('off', 'sldrt:blkgui:boardnotonlist');
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_streamio'));
warning(w);
```



*To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.*



Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up Model

```
clear  
close all  
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “Packet Input/Output” on page 5-18

Video Input

This example shows how to get video data from system video devices. The example model uses a system camera device to provide video input, for example a standard USB video class device (webcam).

When you run the simulation, the model uses signal generator output to control the region-of-interest (ROI) from the camera that is displayed on the Video Viewer block.

For more information about video input or setting up a system camera device, see the Video Input block.

Run Model in Connected IO Mode

- 1 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Connected IO mode**.
- 2 To start the real-time execution, on the **Desktop Real-Time** tab, click **Run in Real-Time**.

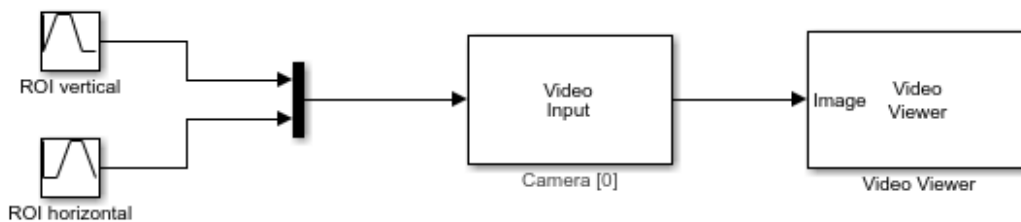
Run Model in Run in Kernel Mode

- 1 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel mode**.
- 2 To start the real-time execution, on the **Desktop Real-Time** tab, click **Run in Real-Time**.

Open the Model

These commands open the model and suppress any warnings about a board not installed.

```
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrteexamples', 'sldrte_xvideoroi'));
```



*To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.*

Copyright 1994-2021 The MathWorks, Inc.

Clean Up Model

```
bdclose('all');
```

Warning: Method 'onBlockDelete' is not defined for class 'vipsopes.VideoViewerBlock' or is removed from MATLAB's search path.

See Also

- Video Input
- “Use Video Input Driver (Mac OS)” on page 4-13

CAN Input/Output

This example shows how to transfer data through CAN bus. The model sends data within one computer, from one CAN channel to another. The two CAN channels can be either virtual channels or physical channels on a dual-channel CAN device. Two different CAN messages using different message identifiers are being transmitted. You can modify the model to communicate between two computers by splitting this model into its send and receive parts and running the models on two computers. The yellow blocks are used to send the data, the blue blocks are used to receive the data.

Note: This model runs on Microsoft Windows only.

Run Model in Connected IO Mode

- 1 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 2 To start the real-time execution, on the **Simulation** tab, click **Run**.

Run Model in Run in Kernel Mode

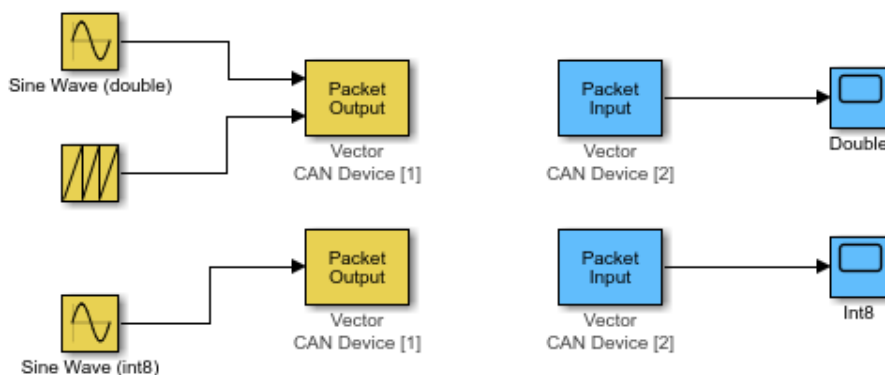
- 1 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 2 To start the real-time execution, click **Run in Real Time**.

The model builds, connects to Simulink in **Run in Kernel** mode, and starts.

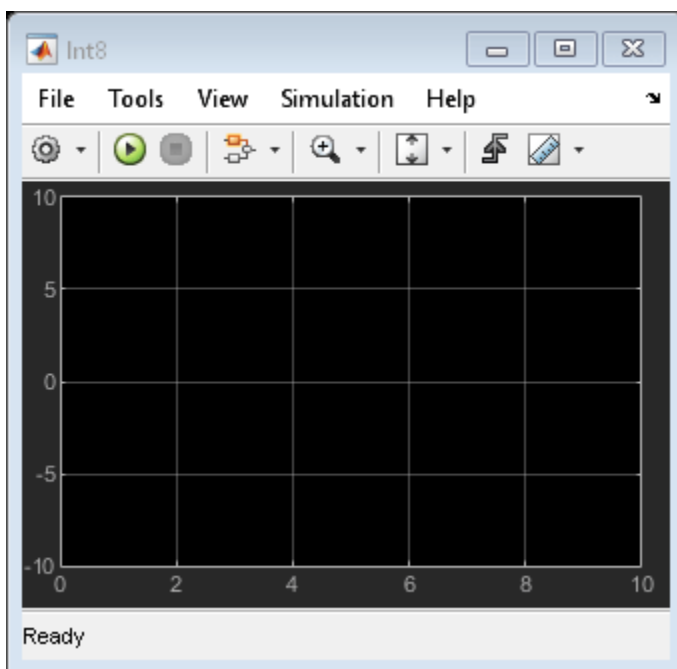
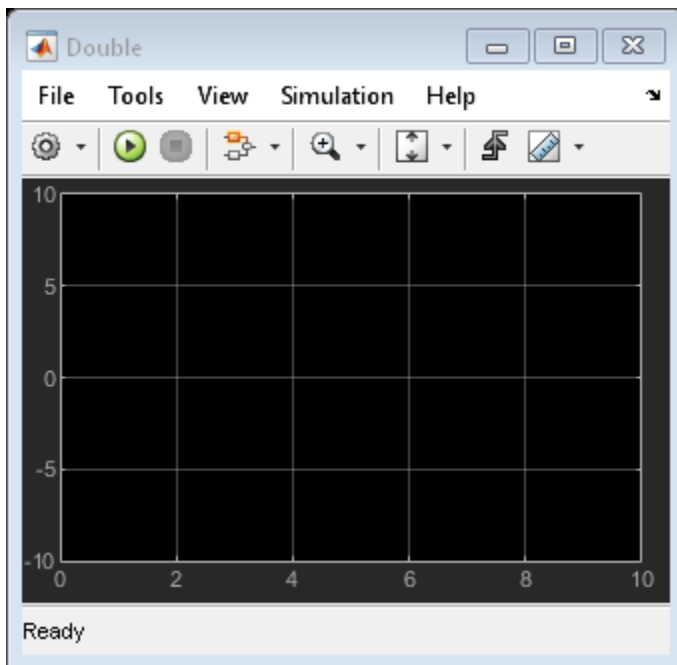
Open the Model

These commands open the model and suppress warning about board not installed.

```
w = warning('off', 'sldrt:blkgui:boardnotonlist');
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_canio'));
warning(w);
```



To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.



Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up Model

```
clear  
close all  
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “CAN Input/Output with Vehicle Network Toolbox” on page 5-29

CAN Input/Output with Vehicle Network Toolbox

This example shows how to transfer data through CAN bus, utilizing the CAN_MESSAGE data type and the CAN Pack and CAN Unpack blocks available in Vehicle Network Toolbox™ block library. The CAN_MESSAGE data type can be directly processed by Simulink Desktop Real-Time™ blocks.

The example also shows the difference in data transfer capability between the CAN and CAN FD protocols. The CAN protocol maximum data size is 8 bytes. This protocol is able to send eight double-precision values, the values need to be scaled and converted to the uint8 data type, losing some precision. The CAN FD protocol maximum data size is 64 bytes. This protocol is capable of transferring eight double-precision values in their native format without any conversion and precision loss.

The model sends data within one computer, from one virtual CAN channel to another. The two CAN channels can be either virtual channels or physical channels on a dual-channel CAN device. You can modify the model to communicate between two computers by splitting this model into its send and receive parts and running the models on two computers.

The yellow blocks are used to send the data. The blue blocks are used to receive the data.

Note: This model runs on Microsoft Windows only.

Run Model in Connected IO Mode

- 1 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 2 To start the real-time execution, click **Run in Real-Time**.

Run Model in Run in Kernel Mode

- 1 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 2 To start the real-time execution, click **Run in Real Time**.

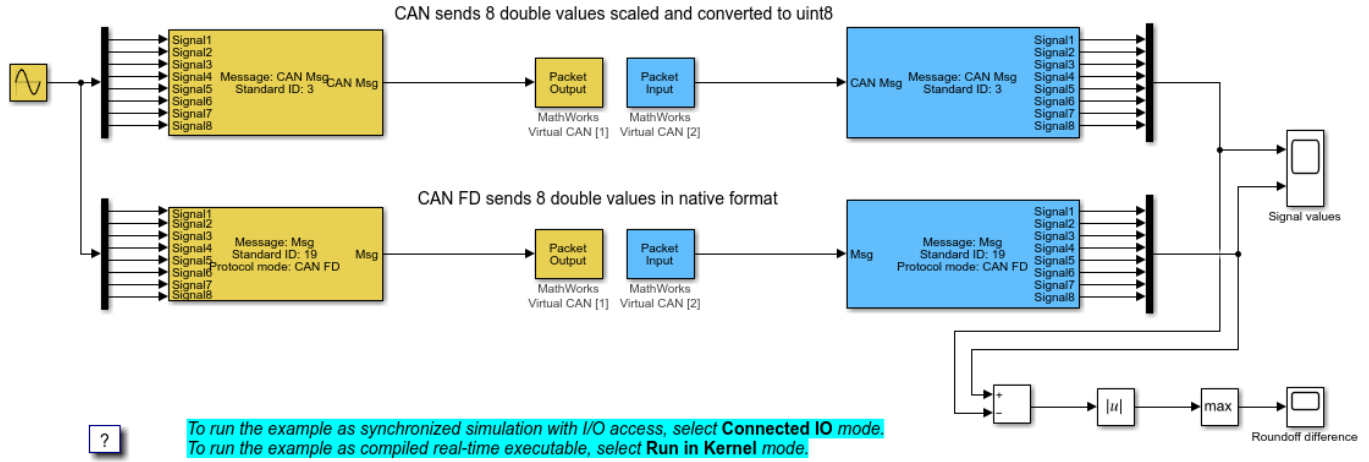
The model builds, connects to Simulink in **Run in Kernel** mode, and starts.

Open the Model

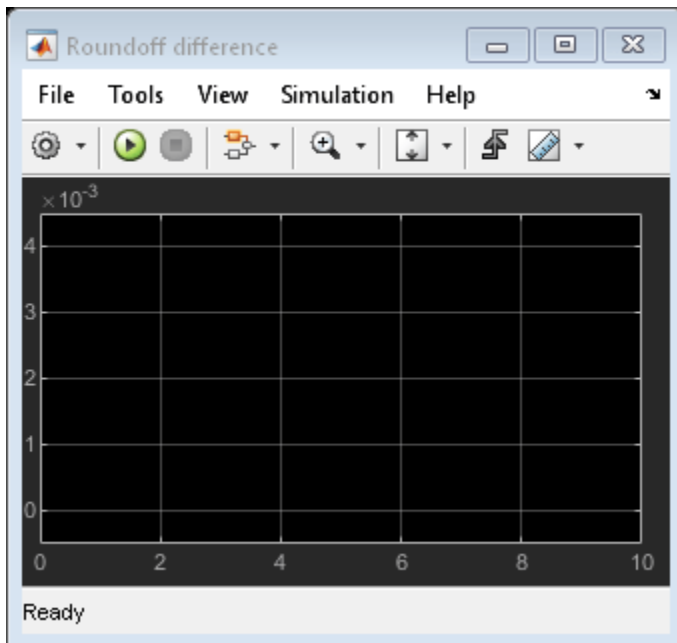
These commands open the model and suppress warning about board not installed.

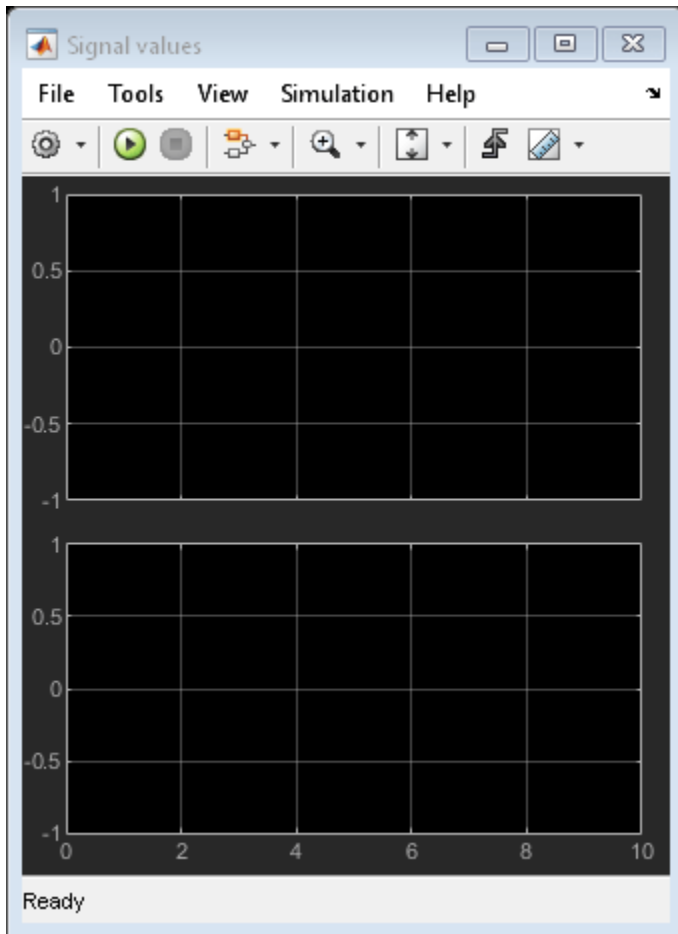
```
w = warning('off', 'sldrt:blkgui:boardnotonlist');
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_canmessage'));
warning(w);
```

5 Simulink Desktop Real-Time Examples



Copyright 1994-2021 The MathWorks, Inc.





Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up Model

```
clear  
close all  
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2
- “CAN Input/Output” on page 5-26

Execution Time Measurement and Block Profiling

This example shows how to analyze model execution performance in Simulink Desktop Real-Time™. The example is a multirate multi-tasking model that performs a time-intensive operation of matrix multiplication and finding the minimum in the resulting matrix product. This is being done for two different matrix sizes at two different sample rates.

The task-level performance information is returned by the Execution Time block. The first output port shows the total time of execution of each base rate step of the entire model. The second output port shows time of execution of each task.

To further refine the performance analysis, one of the two tasks is instrumented to obtain block-level information. The added Timestamp blocks capture the timestamps of input and output signals of blocks that are to be investigated. By subtracting the timestamps for input and output signals, execution times of the blocks are obtained.

Note: This example must be run in **Run in Kernel** Mode and requires Simulink Coder™.

- 1 To switch to **Run in Kernel mode**, on the ***Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 2 To start the real-time execution, click **Run in Real Time**.

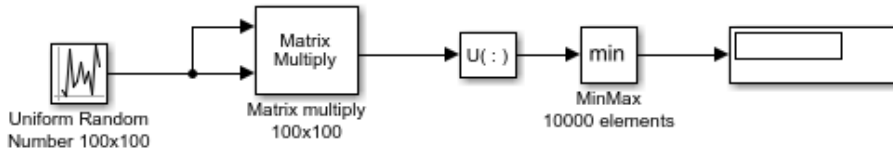
The model builds, connects to Simulink in **Run in Kernel** mode, and starts.

Open the Model

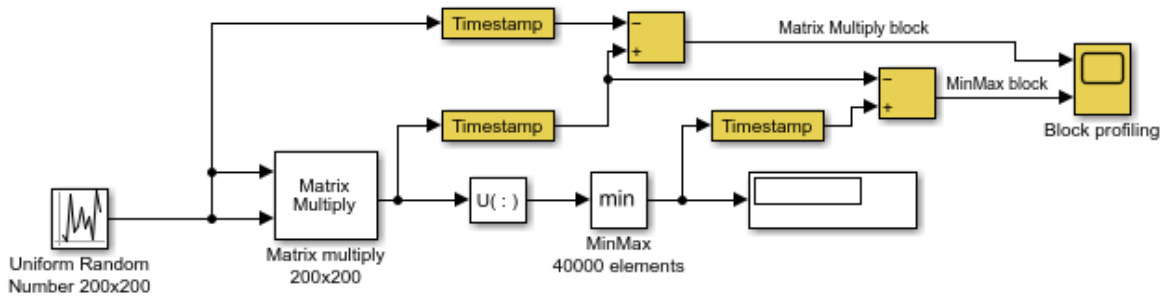
```
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_profiling'));
```


Execution Time Measurement and Block Profiling

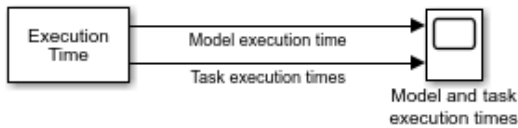
Task 1: sample rate 0.02 s, matrix size 100x100



Task 2: sample rate 0.1 s, matrix size 200x200, instrumented for block profiling



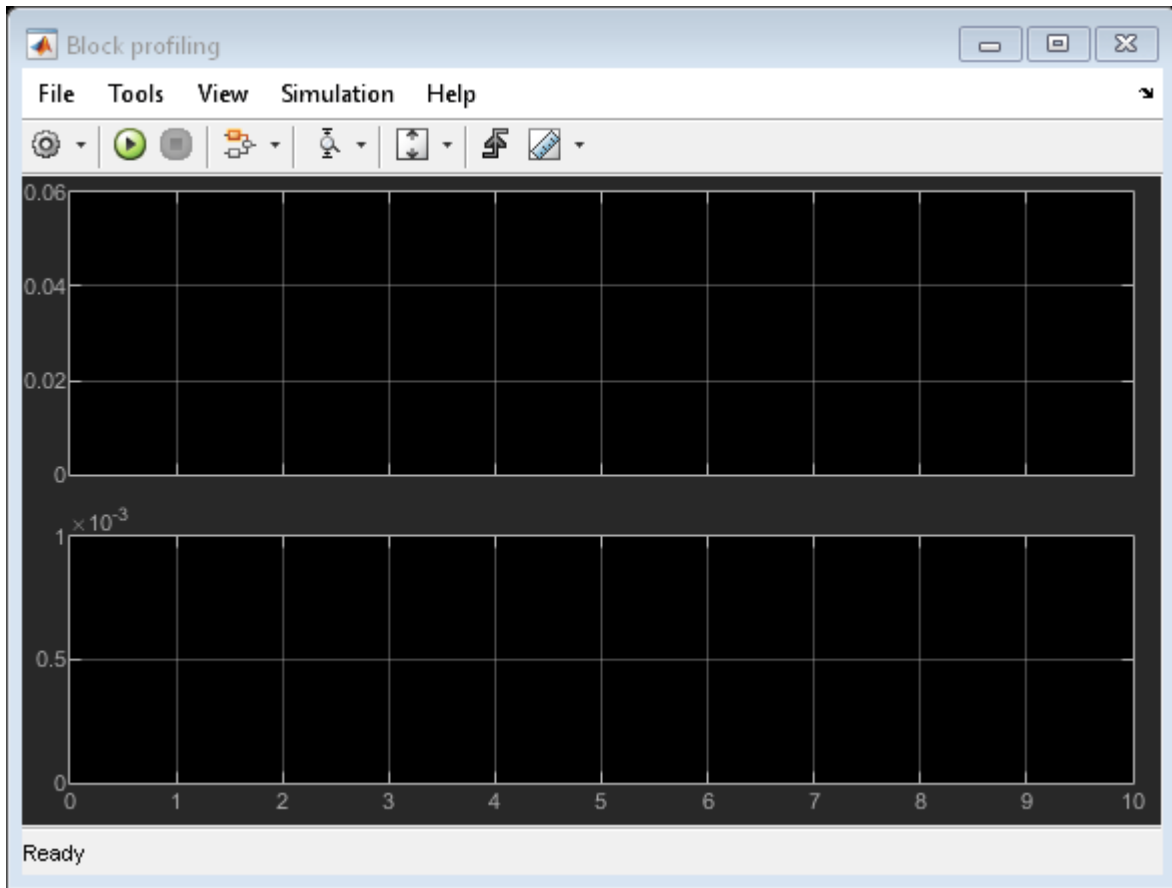
Model and task execution time measurement

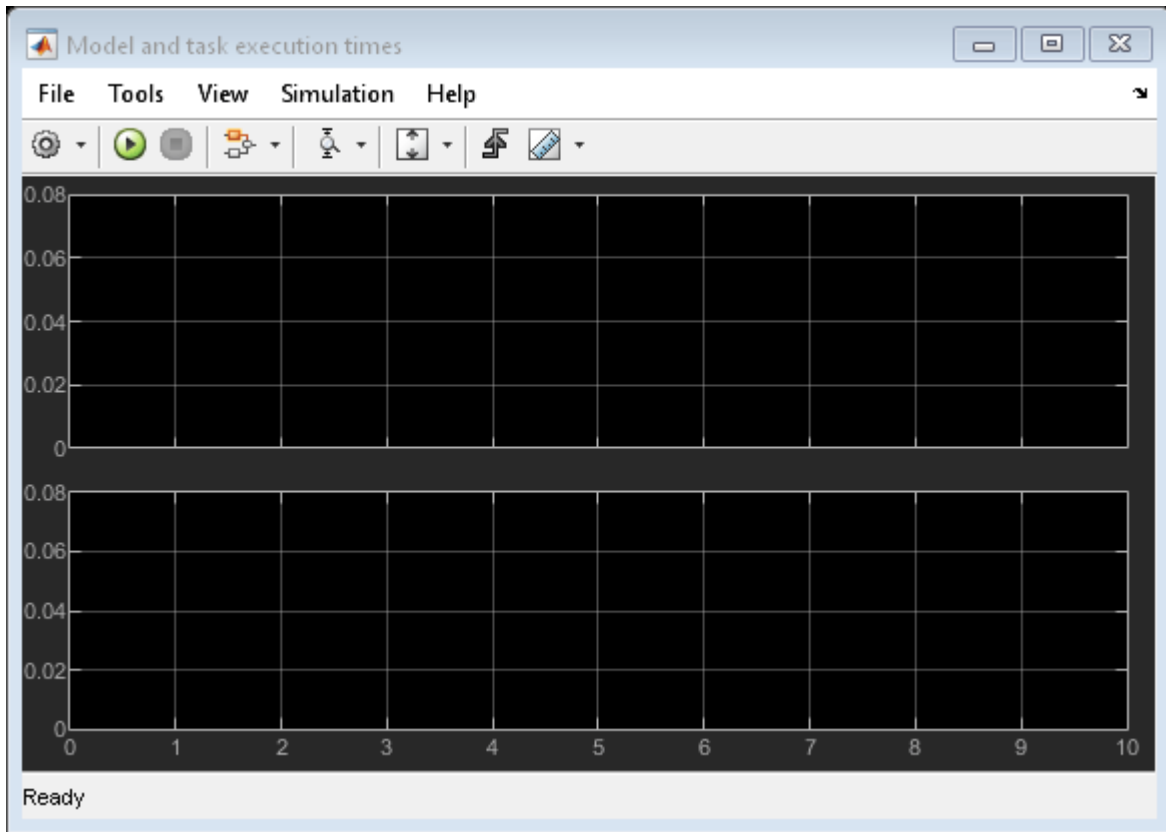


This example can be run in Run in Kernel mode only.



Copyright 1994-2021 The MathWorks, Inc.





Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up Model

```
clear  
close all  
bdclose all
```

See Also

- "Prepare for Real-Time Execution" on page 3-2
- "Inspect Simulink® Desktop Real-Time™ Signals with Simulation Data Inspector" on page 3-32

Apply Simulink Desktop Real-Time Model Templates to Create Real-Time Models

Starting from the model template for Simulink Desktop Real-Time™ provides a new model that has configuration parameters set up for building a real-time application. This example shows how to use the Simulink Desktop Real-Time template for a new Simulink model that is configured for **Connected IO** mode or **Run in Kernel** mode.

To see the Simulink Desktop Real-Time commands for each operation in this example, view the example code.

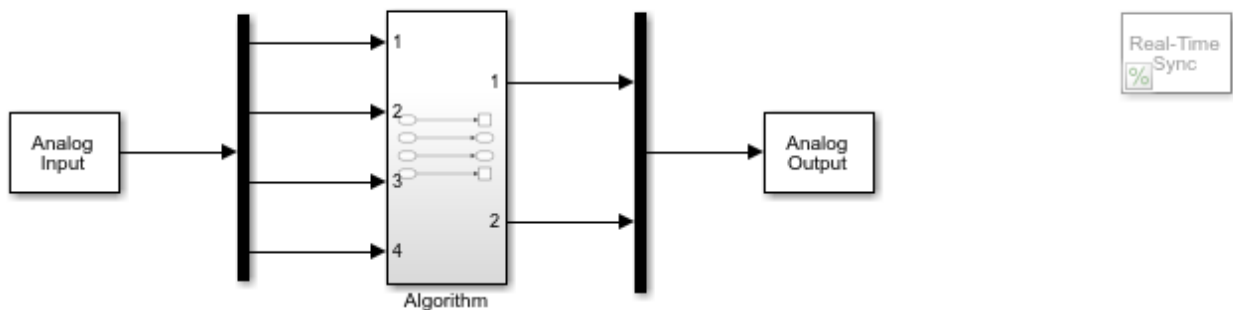
Create a Simulink Desktop Real-Time Model from Template

To create this model from the Simulink start page, in the Command Window, type:

```
simulink
```

To create a model that is configured for **Connected IO** mode, select the Simulink Desktop Real-Time **Connected IO** mode template from the start page, and create the `exampleSldrAppConnectedIO` model. Or, in the Command Window, use the `Simulink.createFromTemplate` command.

Simulink Desktop Real-Time Connected IO Mode

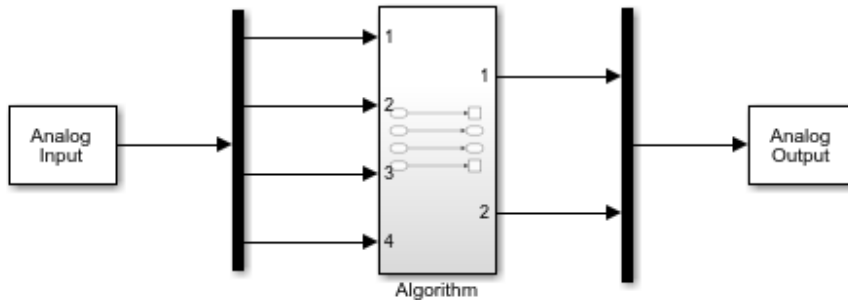


Tips for maximum performance:

1. Both fixed-step and variable-step solvers can be used in Connected IO mode.
2. All I/O blocks perform real-time synchronization. Use the Real-Time Synchronization block only if no I/O block is used.
3. Use a single block that reads or writes all channels of given type, rather than multiple blocks for one channel each.

To create a model that is configured for **Run in Kernel** mode, select the Simulink Desktop Real-Time **Run in Kernel** mode template from the start page, and create the `exampleSldrAppRunInKernel` model. Or, in the Command Window, use the `Simulink.createFromTemplate` command.

Simulink Desktop Real-Time Run in Kernel Mode



Tips for maximum performance:

1. It is not necessary to use the Real-Time Synchronization block in Run in Kernel mode. The block performs no operation.
2. It is not useful to use the Missed Ticks port in Run in Kernel mode. The port always outputs zero.
3. Use a single block that reads or writes all channels of given type, rather than multiple blocks for one channel each.

Tips for Maximum Performance

These are some tips to help you get the most performance from the models that you create from these model templates.

For model that is configured for **Connected IO** mode:

- Both fixed-step and variable-step solvers can be used in **Connected IO** mode.
- All I/O blocks perform real-time synchronization. Use the Real-Time Synchronization block only if no I/O block is used.
- Use a single block that reads or writes all channels of given type, rather than multiple blocks for one channel each.

For model that is configured for **Run in Kernel** mode:

- It is not necessary to use the Real-Time Synchronization block in **Run in Kernel** mode. The block performs no operation.
- It is not useful to use the Missed Ticks port in **Run in Kernel** mode. The port always outputs zero.
- Use a single block that reads or writes all channels of given type, rather than multiple blocks for one channel each.

More Information

- “Create a Real-Time Application” on page 3-4
- “Real-Time Execution in Connected IO Mode” on page 1-3
- “Real-Time Execution in Run in Kernel Mode” on page 1-5

UDP String Data and Message Handling

This example shows how to transfer text data by using the UDP communication protocol and shows how to store the incoming messages in a FIFO queue.

The model receives randomly distributed text messages containing textual color description by using the Stream Input block. The model converts the textual color description into corresponding numeric color code. A FIFO queue stores the color codes. The model sequentially displays the received colors at a sample rate that is perceptible to the human eye.

The Message generator subsystem simulates a remote device by randomly sending color text messages to the model. These messages provide input data for the example.

Run Model in Connected IO Mode

- 1 To switch to **Connected IO** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 2 To start the real-time execution, click **Run in Real-Time**.

Run Model in Run in Kernel Mode

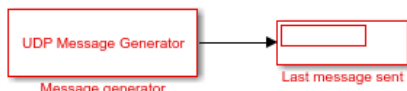
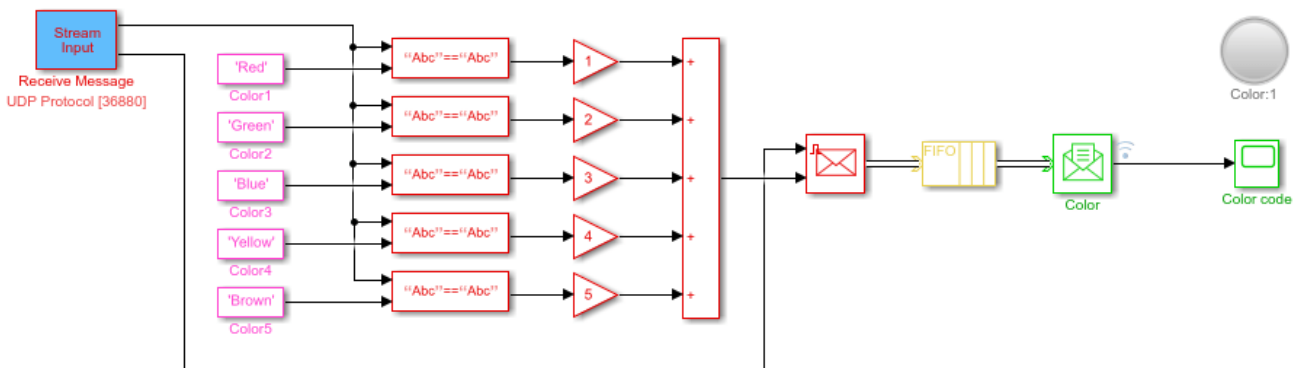
- 1 To switch to **Run in Kernel** mode if needed, on the **Desktop Real-Time** tab, select **Mode > Run in Kernel**.
- 2 To start the real-time execution, click **Run in Real-Time**.

The model builds, connects to Simulink in **Run in Kernel** mode, and starts.

Open the Model

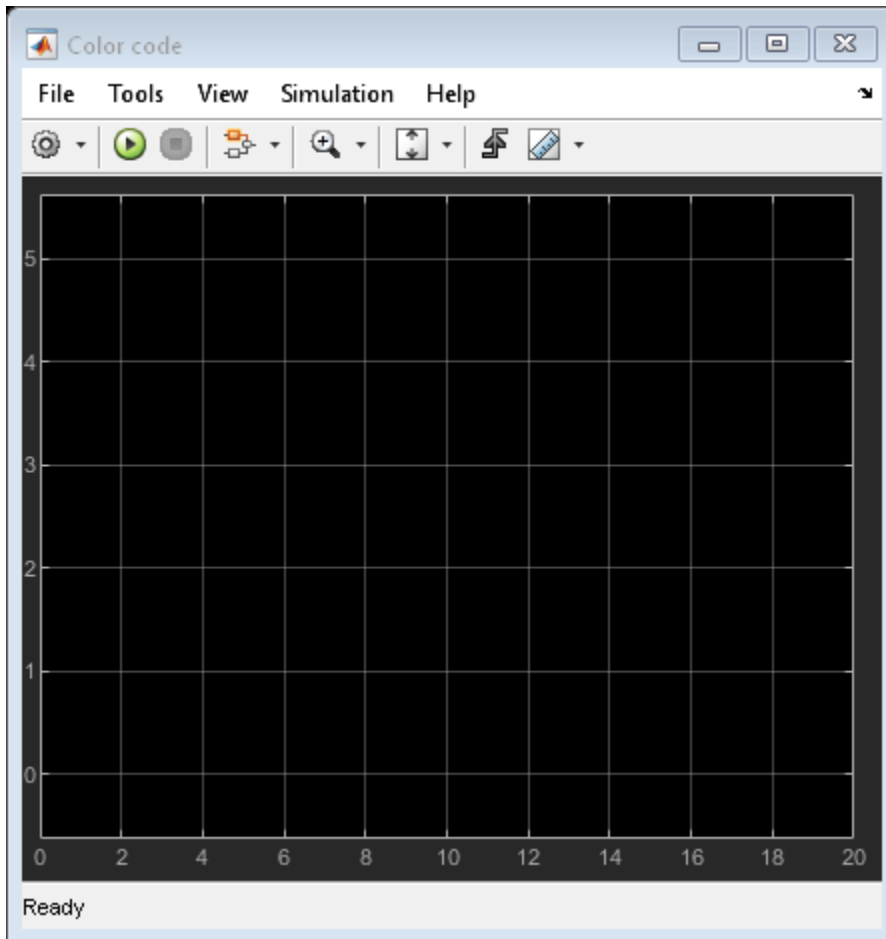
These commands open the model and suppress warning about board not installed.

```
w = warning('off', 'sldrt:blkgui:boardnotonlist');
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_stringmessage'));
warning(w);
```



To run the example as synchronized simulation with I/O access, select **Connected IO mode**.
To run the example as compiled real-time executable, select **Run in Kernel mode**.





Close Open Scopes

```
close_system(find_system(gcs , 'BlockType', 'Scope'));
```

Clean Up Model

```
clear  
close all  
bdclose all
```

See Also

- “Prepare for Real-Time Execution” on page 3-2

Sync Models by Using Arduino Connected I/O Board

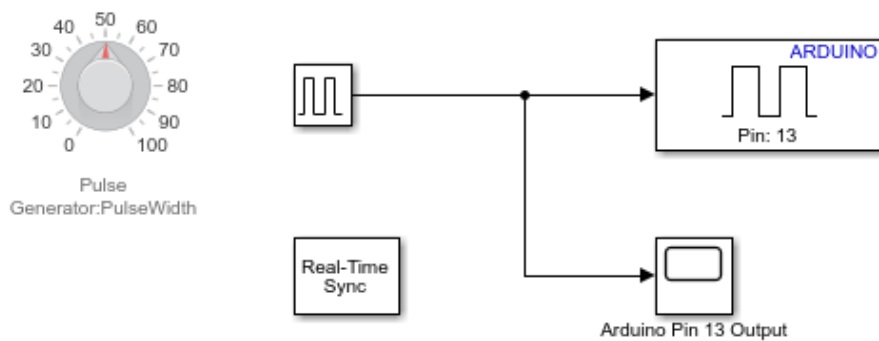
This example shows how to synchronize models using an Arduino Connected I/O board with real time and using Simulink Desktop Real-Time running in **Connected IO** mode.

The model sends software-generated pulse-width modulation signal to Arduino pin 13 where it can be observed by an oscilloscope or visually by observing the LED connected to pin 13. The duty of the signal can be controlled by the attached knob control.

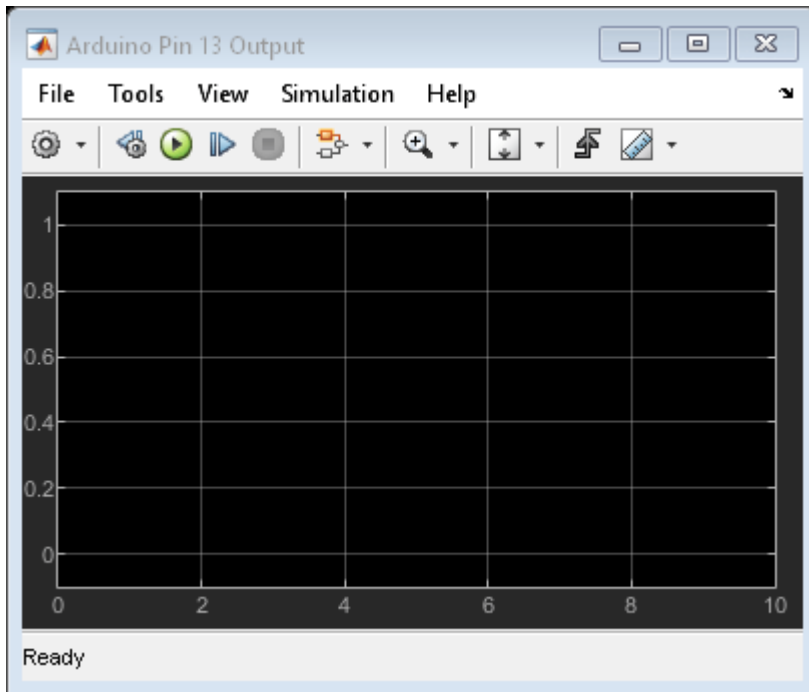
Open the Model

Open the model `sldrtex_arduino`. In the MATLAB Command Window, type:

```
open_system(fullfile(matlabroot, 'toolbox', 'sldrt', 'sldrtexamples', 'sldrtex_arduino'));
```



*This example can be run in **Connected IO** mode only.
This example requires Simulink Support Package for Arduino Hardware.*



Run the Model

To run this model, use this setup:

- 1 The model is currently configured for Arduino Mega 2560. If necessary, change the configuration to match your Arduino model.
- 2 To change to **Connected IO** mode (required), on the **Desktop Real-Time** tab, select **Mode > Connected IO**.
- 3 To start real-time execution, click **Run in Real-Time**.

Note: This model requires Simulink Support Package for Arduino Hardware.

Troubleshooting

Solutions have been worked out for some common errors and problems that can occur when you are using Simulink Desktop Real-Time software. For more information, see Simulink Desktop Real-Time — MATLAB Answers.

- “Troubleshoot Missing Desktop Real-Time Tab” on page 6-2
- “Troubleshoot sldrttext Incorrect Version Error” on page 6-3
- “Troubleshoot Delayed or Missing Scope Output” on page 6-4
- “Troubleshoot Signals Not Plotted in Scope Blocks” on page 6-5
- “Troubleshoot Vendor Software Missing Issues” on page 6-7
- “Troubleshoot Builds of Referenced Models” on page 6-8
- “Troubleshoot Slow or Halted Simulation on Windows” on page 6-9
- “Troubleshoot C++ Standard Template Library (STL) Compilation Errors for Real-Time Application” on page 6-12

Troubleshoot Missing Desktop Real-Time Tab

Where is the **Desktop Real-Time** tab? I do not see this tab in the Simulink editor.

What This Issue Means

From the model configuration, the Simulink editor determines which tabs to display. The editor displays the **Desktop Real-Time** tab for models that are configured for Simulink Desktop Real-Time.

Try This Workaround

To configure your model for Simulink Desktop Real-Time, in Simulink Editor, from the **Apps** tab, click **Desktop Real-Time**.

This operation changes the code generation target to `sldrt.tlc` for the model. After changing the configuration, the Simulink editor displays the **Desktop Real-Time** tab for the model.

See Also

More About

- “Configure a Model for Simulink Desktop Real-Time” on page 3-10

Troubleshoot sldrtext Incorrect Version Error

When I try to run a Simulink Desktop Real-Time model in external mode, the Simulation Errors dialog box displays:

```
Error occurred while executing External Mode MEX-file 'sldrtext': Incorrect version
```

What This Issue Means

This message indicates that the Simulink Desktop Real-Time kernel could be out-of-date.

Try This Workaround

To check the real-time kernel version, in the Command Window, type `rtwho`. If the real-time kernel is out-of-date, you see a message similar to:

```
Incorrect kernel version is installed.  
Use 'sldrtkernel -setup' to install the correct version.
```

To resolve this issue, install the current version of the real-time kernel. :

See Also

More About

- “Install Real-Time Kernel” on page 2-4

Troubleshoot Delayed or Missing Scope Output

When simulating a Simulink Desktop Real-Time model in external mode run, I notice slow updates of Scope blocks or failure to plot data in Scope blocks.

What This Issue Means

Slow updates of Scope blocks or failure to plot data in Scope blocks could indicate that the real-time application sample time is near the lower threshold for the computer and I/O hardware.

Note The sample time is set in the **Fixed step size (fundamental sample time)** field in the Configuration Parameters **Solver** pane. The **Fixed step size** field appears only when **Type** is set to **Fixed-step**.

Plotting data has a lower priority than executing the application. A small sample time allows the application to run but can leave insufficient resources for plotting. If the sample time is so small that the application itself cannot run, an error message is displayed and real-time execution is terminated.

Try This Workaround

To check the sample time, select a larger sample time for your application. Change the sample time of any I/O drivers to be the same as the new application sample time or to an integer multiple of that time. Then rebuild the model, connect to the target, and restart the real-time application.

As required, iterate changing the application sample time until scope output appears. For example, start with a sample time of 0.01 seconds, and confirm that your system runs and plots are displayed. Then decrease the sample time until you can display scopes and meet your accuracy and response time requirements. After changing the application sample time, update the I/O driver sample time and rebuild the application .

See Also

More About

- “Prepare for Real-Time Execution” on page 3-2
- “Real-Time Execution in Run in Kernel Mode” on page 1-5

Troubleshoot Signals Not Plotted in Scope Blocks

When I simulate a Simulink Desktop Real-Time model in **Run in Kernel** mode, I do not see signals plotted in the Scope blocks.

What This Issue Means

Missing signal plots in Scope blocks during simulation could indicate a model configuration issue or a sample rate issue.

Try This Workaround

To check whether there is a model configuration issue or a sample rate issue, try these workarounds.

Check the Model Configuration

Before you execute your application in **Run in Kernel** mode, you must specify data to plot in a Simulink Scope block.

- 1 Open the model.
- 2 In the Simulink Editor, on the **Desktop Real Time** tab, click **Prepare > Control Panel**.
- 3 In the External Mode Control Panel, click **Signals & Triggering** and select one or more signals for capture (designated with X) in the External Signal & Triggering dialog box.
- 4 Set **Duration * Fixed Step Size** close to or less than the X range in the Scope block.
- 5 Select required mode (one-shot or normal).
- 6 Configure signal levels to allow triggering.
- 7 Set the Y range on Simulink Scope block axes large enough to span the signal amplitude.
- 8 Set the X range large enough to provide required time resolution.
- 9 Set **Arm when connect to target** in the External Signal & Triggering dialog box or **Arm Trigger** in the External Mode Control Panel.
- 10 On the **Desktop Real Time** tab, click **Mode > Run in Kernel**.
- 11 On the **Desktop Real Time** tab, click **Run in Real Time**.

Simulink builds the real-time application, connects to the kernel, and starts the real-time simulation. The Scope displays the signals.

Run an Example Model or Reduce the Sample Rate

Try running one of the example models or run your model at a slower sample rate.

If you can see signals plotted for an example model or for your model at a slower sample rate, your system cannot transfer data back to MATLAB for plotting in the CPU time available between sample intervals for your model and the original sample rate.

Select the fastest sample rate that allows your model to run and plot signals.

See Also

More About

- “Prepare for Real-Time Execution” on page 3-2
- “Real-Time Execution in Run in Kernel Mode” on page 1-5

Troubleshoot Vendor Software Missing Issues

I use the Simulink Desktop Real-Time software drivers for my I/O modules. I see warning and error messages about missing software during model builds.

What This Issue Means

Simulink Desktop Real-Time provides dedicated drivers for I/O modules. Some vendors can require the installation of vendor-specific software, even if you do not use their software. A warning or error message during the build or during execution can indicate this software installation requirement. Or, the I/O module fails to initialize.

Try This Workaround

To resolve warning or error messages issue from third-party vendor software, refer to the vendor documentation for vendor-specific requirements. If the hardware requires the installation of vendor software, install the vendor software on your computer.

See Also

More About

- “Software Components” on page 2-2

External Websites

- Hardware Support from Simulink Desktop Real-Time

Troubleshoot Builds of Referenced Models

During referenced model builds, I get warning and error messages for many Simulink Desktop Real-Time blocks.

What This Issue Means

Simulink Desktop Real-Time supports a limited list of blocks in referenced models for model builds.

Try This Workaround

These blocks are not supported in referenced models. Place these blocks at the top level of the model.

- Analog Input
- Analog Output
- Counter Input
- Encoder Input
- Frequency Output
- Digital Input
- Digital Output
- Packet Input
- Packet Output
- Stream Input
- Stream Output
- Other Input
- Other Output
- Real-Time Sync

These blocks are supported in referenced models.

- Execution Time
- Timestamp

See Also

More About

- “Prepare for Real-Time Execution” on page 3-2

Troubleshoot Slow or Halted Simulation on Windows

On my Windows system, during software installation and when I try to run real-time simulation, I see this warning message:

```
The Hyper-V hypervisor has been detected. The Simulink Desktop Real-Time kernel cannot run in the presence of the hypervisor. Please disable the Hyper-V operating system component before attempting to use Simulink Desktop Real-Time.
```

Or, on releases previous to R2022a, I see slow or halted simulation on my Windows system. Simulation fails with an error message:

```
Warning: The "Real-Time Synchronization" block has timed out while trying to synchronize to real-time kernel.
```

What This Issue Means

This message indicates that the system could not load and start the kernel due to Microsoft® virtualization.

Try This Workaround

To identify the cause of the issue, in the Command Window, type:

```
!systeminfo
```

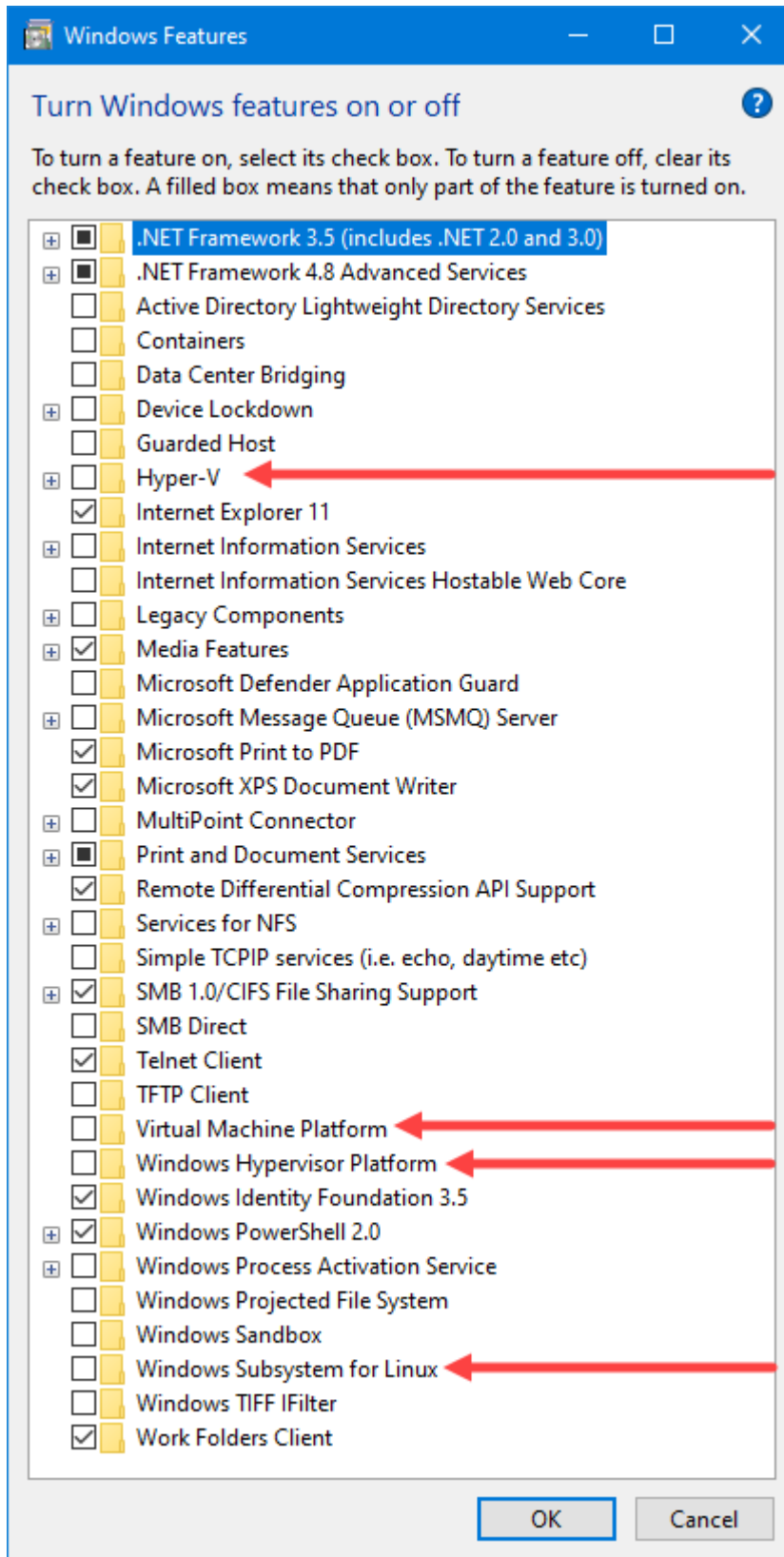
Examine the system information from these commands. Check whether the last line is:

```
Hyper-V Requirements:
```

```
  A hypervisor has been detected. Features required for Hyper-V will not be displayed.
```

The Windows Hyper-V feature and other Microsoft virtualization features are not compatible with the Simulink Desktop Real-Time kernel. You cannot use Microsoft virtualization with the real-time kernel. To disable Hyper-V and other Microsoft virtualization features:

- 1 From the **Start** menu, search for **Turn Windows features on or off**.
- 2 Select **Turn Windows features on or off**.
- 3 Clear the boxes for features:
 - **Hyper-V**
 - **Virtual Machine Platform**
 - **Windows Hypervisor Platform**
 - **Windows Subsystem for Linux**



- 4 After the uninstall process completes, restart the computer as prompted.

See Also

More About

- “Software Components” on page 2-2

External Websites

- System Requirements
- Hyper-V feature

Troubleshoot C++ Standard Template Library (STL) Compilation Errors for Real-Time Application

To include a C++ project, I wrap the source into an S-Function block that is compatible with the Simulink Desktop Real-Time code generation target, `sldrt.tlc`. The C++ project uses the C++11 Standard Template Library (STL) interfaces, such as `std::vector`, `std::stack`, and `std::complex`. I can successfully compile the project in **Connected IO** mode. But, when I compile in **Run in Kernel** mode, I get compilation errors such as:

```
<complex> file not found  
<vector> file not found
```

What This Issue Means

The compiler generates these errors because the C++ standard template library is not compatible with real-time code.

In real-time code, there is a requirement to always finish a single time step on time, before the next one is due to start. This requirement produces time-deterministic behavior for the real-time code and produces more or less fixed execution time.

By comparison, the C++ standard template library frequently uses features that are not time-deterministic. The most notable examples are dynamic memory allocation and exceptions. While dynamic memory allocation is (with some limitations) supported by the real-time kernel in a real-time deterministic way, exceptions are not. So, it is not possible to use code that can throw exceptions.

This issue causes most of the STL headers to be unavailable with the Simulink Desktop Real-Time code generation target. These interfaces throw exceptions. You cannot run code that uses the STL in the real-time kernel.

Try This Workaround

To eliminate the compiler errors, modify your project not to use the C++ standard template library.

If you are not able to modify your project not to use the C++ standard template library, you can use the Simulink Desktop Real-Time code generation target in **Connected IO** mode.

For **Connected IO** mode, compile the S-function as you would for Simulink. Then, run the model. The real-time requirements cannot be enforced, but the real-time misses are reported as they occur.