Simulink® Desktop Real-Time™

User's Guide

MATLAB®

R2016b

MathWorks®

How to Contact MathWorks

🖳	Latest news:	www.mathworks.com
	Sales and services:	www.mathworks.com/sales_and_services
	User community:	www.mathworks.com/matlabcentral
	Technical support:	www.mathworks.com/support/contact_us
☎	Phone:	508-647-7000

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

Simulink® Desktop Real-Time™ User's Guide

Trademarks

Patents

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)

Contents

1 Getting Started

2 Installation and Configuration

3 Basic Procedures

4 Boards, Blocks, and Drivers

5 Troubleshooting

A Custom I/O Driver Blocks

Getting Started

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 up to a 1 kHz sample rate with Simulink, and up to 20 kHz 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 Normal mode
- Real-time performance approaching a 20 kHz sample rate in Simulink External 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

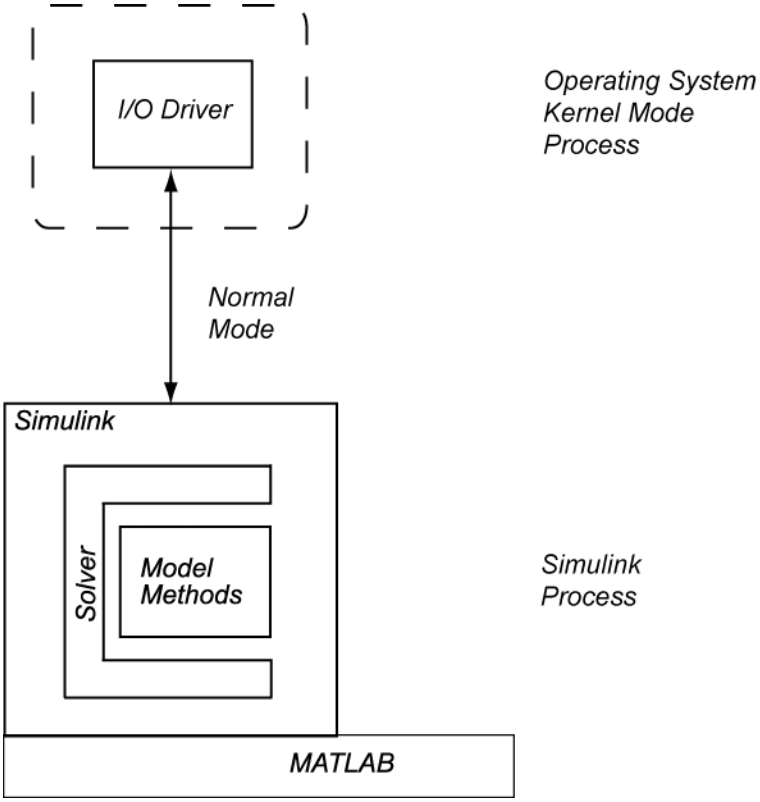
Real-Time Execution in Normal Mode

Simulink Desktop Real-Time extends Simulink normal mode to run in real time.

The simulation algorithm for a non-real-time Simulink normal mode 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. However, it is not synchronized with a real-time clock and cannot easily be used to operate real-time hardware.

You can synchronize a Simulink model with a real-time clock using Simulink Desktop Real-Time I/O blocks. In real-time normal mode, Simulink executes the simulation algorithm, while a separate operating system kernel mode process runs I/O drivers for the I/O blocks. Both the Simulink process and the kernel mode process run on the host machine, 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 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.



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, the step size is set in the **Fixed step size** box from the Configuration Parameters dialog box. For a variable-step simulation, the step size is determined automatically by Simulink or by the **Min Step Size** attribute.

In real-time normal mode, at each sample interval Simulink evaluates each real-time block and writes the input data into a buffer 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 might 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. This situation is marked 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, but 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. This situation is marked 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 per algorithm step may vary during simulation. If Simulink execution does not reach the `Real-Time Sync` or I/O blocks in time to synchronize with the tick, the number of missed ticks jumps to some number, then decreases to zero as Simulink catches up with the kernel. As with process contention, this situation is marked by a sawtooth-like shape on a `Scope` connected to the `Missed Ticks` port.

Real-Time Execution in Accelerator Mode

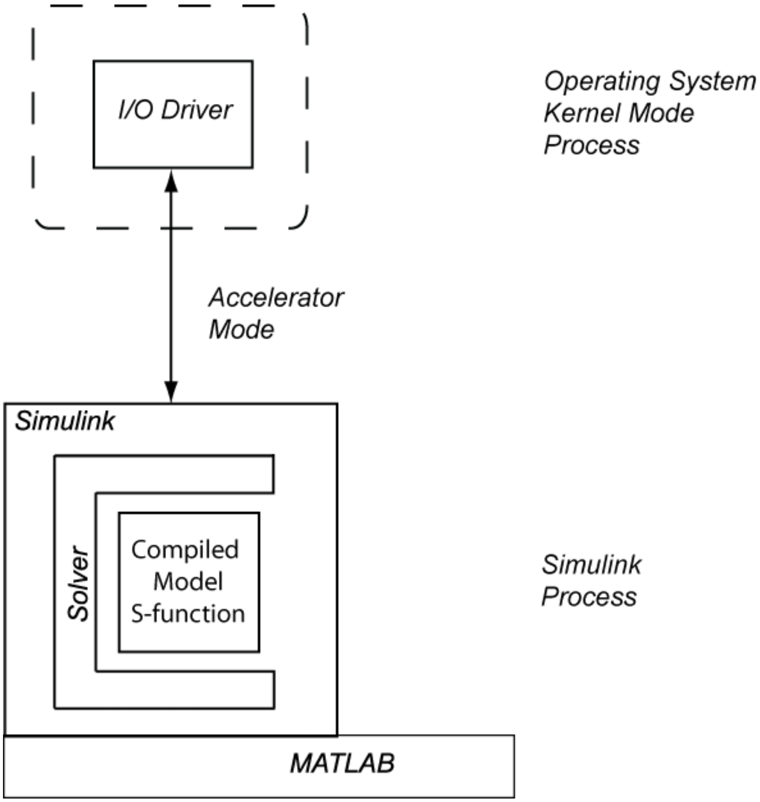
Simulink Desktop Real-Time extends Simulink accelerator mode to run in real time.

The simulation algorithm for a Simulink accelerator mode model is compiled to a MEX file S-function. As with normal mode, the S-function runs in the Simulink process. It is not synchronized with a real-time clock and cannot easily be used to operate real-time hardware.

You can synchronize an accelerator mode model with a real-time clock using Simulink Desktop Real-Time I/O blocks. In real-time accelerator mode, the simulation algorithm S-function runs in the Simulink process, while a separate operating system kernel mode process runs I/O drivers for the I/O blocks. Both the Simulink process and the kernel mode process run on the host machine, 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 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, the step size is set in the **Fixed step size** box from the Configuration Parameters dialog box. For a variable-step simulation, the step size is determined automatically by Simulink or by the **Min Step Size** attribute.

In real-time accelerator mode, at each sample interval Simulink evaluates each real-time block and writes the input data into a buffer 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 might 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. This situation is marked 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, but 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. This situation is marked 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 per algorithm step may vary during simulation. If Simulink execution does not reach the `Real-Time Sync` or I/O blocks in time to synchronize with the tick, the number of missed ticks jumps to some number, then decreases to zero as Simulink catches up with the kernel. As with process contention, this situation is marked by a sawtooth-like shape on a `Scope` connected to the `Missed Ticks` port.

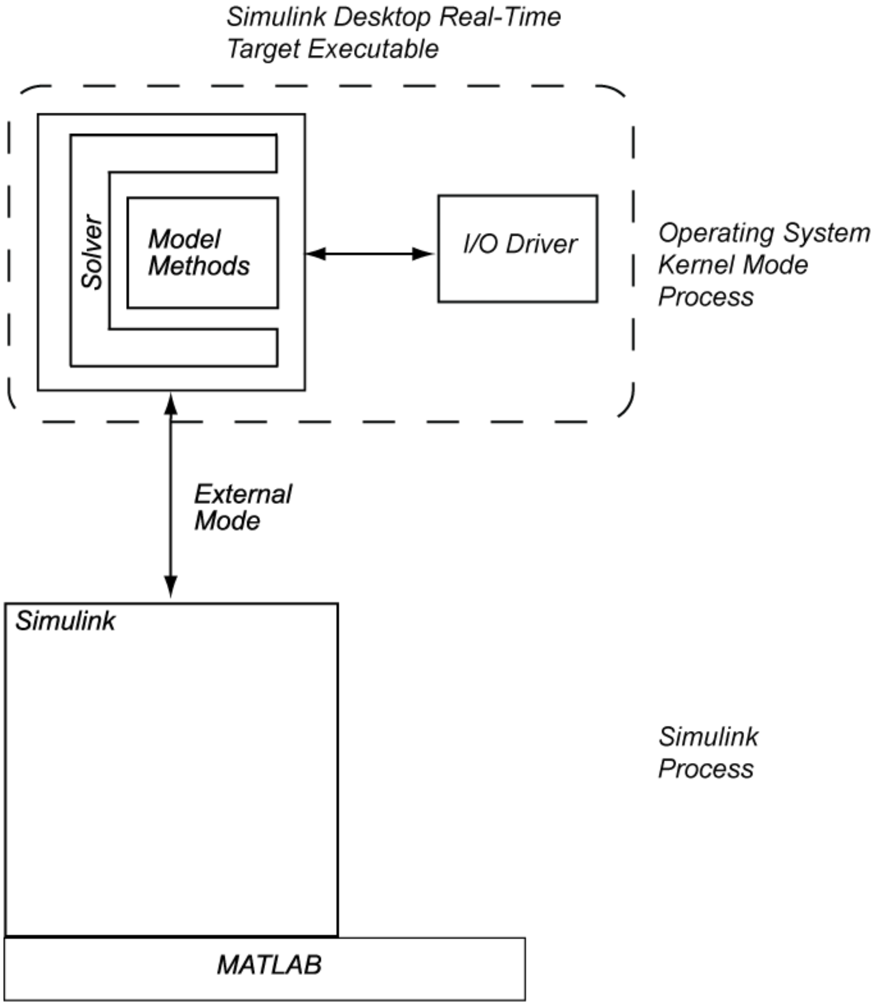
Real-Time Execution in External Mode

A higher-performance alternative to real-time normal mode is real-time external mode. In this mode, you use Simulink Coder to dynamically link generated algorithm code with I/O driver code generated from the I/O blocks. The resulting executable runs in operating system kernel mode on the host computer and exchanges parameter data with Simulink via 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 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. Simulink external mode changes parameters in your real-time application while it is running in real time.

The external 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.

Note: You must use a fixed-step solver in external mode.



In external mode, the real-time application runs in the kernel mode process. Using I/O drivers running in that process to communicate with the hardware, it 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 selected sample interval. Therefore, data transfer runs at a lower priority in the remaining CPU time after model computations are performed while waiting 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, or it can be saved directly to a MAT-file using the data archiving feature of the Simulink external mode.

With data archiving, each buffer of data can be saved to its own MAT-file. The MAT-file names can be automatically incremented, allowing you to capture and automatically store many data buffers. Although points within a buffer are contiguous, the time required to transfer data back to Simulink forces an intermission for data collection until the entire buffer has been transferred and may result in lost sample points between data buffers.

Installation and Configuration

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. C code S-functions are supported by Simulink Coder code generation software.

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: Some of the functions on a board may not be supported by Simulink Desktop Real-Time software. Check the MathWorks® web site for an updated list of supported boards and functions at `www.mathworks.com/hardware-support/simulink-desktop-real-time.html`.

Simulink Coder Software

Simulink Coder code generation software provides the 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 external 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 external mode, the Simulink Desktop Real-Time software does not support the following:
  - Simscape™ or Simscape Driveline™ blocks
  - Blocks that do not run in real time
  - `To File` blocks
- Limitations with Simulink Coder code generation software:
  - In external mode, MATLAB S-functions are not supported.
  - When you use a continuous-time system and generate code for external 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 Distributed Computing 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.

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 of your model, 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 might need servicing.

In this section...

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 making changes.

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 will not run.
4. After installing the kernel, verify that it was installed by typing:

   ```
   rtwho
   ```

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

Once the kernel is installed, you can leave it installed. The kernel remains idle after you have installed it, which allows the operating system to control the execution of standard applications, such as Internet browsers, word processors, the MATLAB environment, and so on. 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:

   ```
   sldrtkernel -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 cancel uninstallation without making changes.

   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, verify that it was uninstalled. Type:

```
rtwho
```

The MATLAB Command Window displays the message:

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

Uninstall the Kernel Using Host Computer Command Line

Uninstalling the MATLAB environment does not uninstall the Simulink Desktop Real-Time kernel. If you have uninstalled the MATLAB environment and need to uninstall the kernel, launch a command window and type:

```
sldrtkernel -uninstall
```

The `sldrtkernel` 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.

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 Simulink normal mode (initial setting), accelerator mode, or Simulink external 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, you should test the installation by running the model `sldrtex_vdp`. If you change your installation, you should 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 launch MATLAB Help, open **Simulink Desktop Real-Time**, and choose **Examples** > **Real-Time Van der Pol Simulation**.

In this section...
“Run the Model sldrtex_vdp” on page 2-7
“Display Status Information” on page 2-10
“Examples Library” on page 2-10

Run the Model sldrtex_vdp

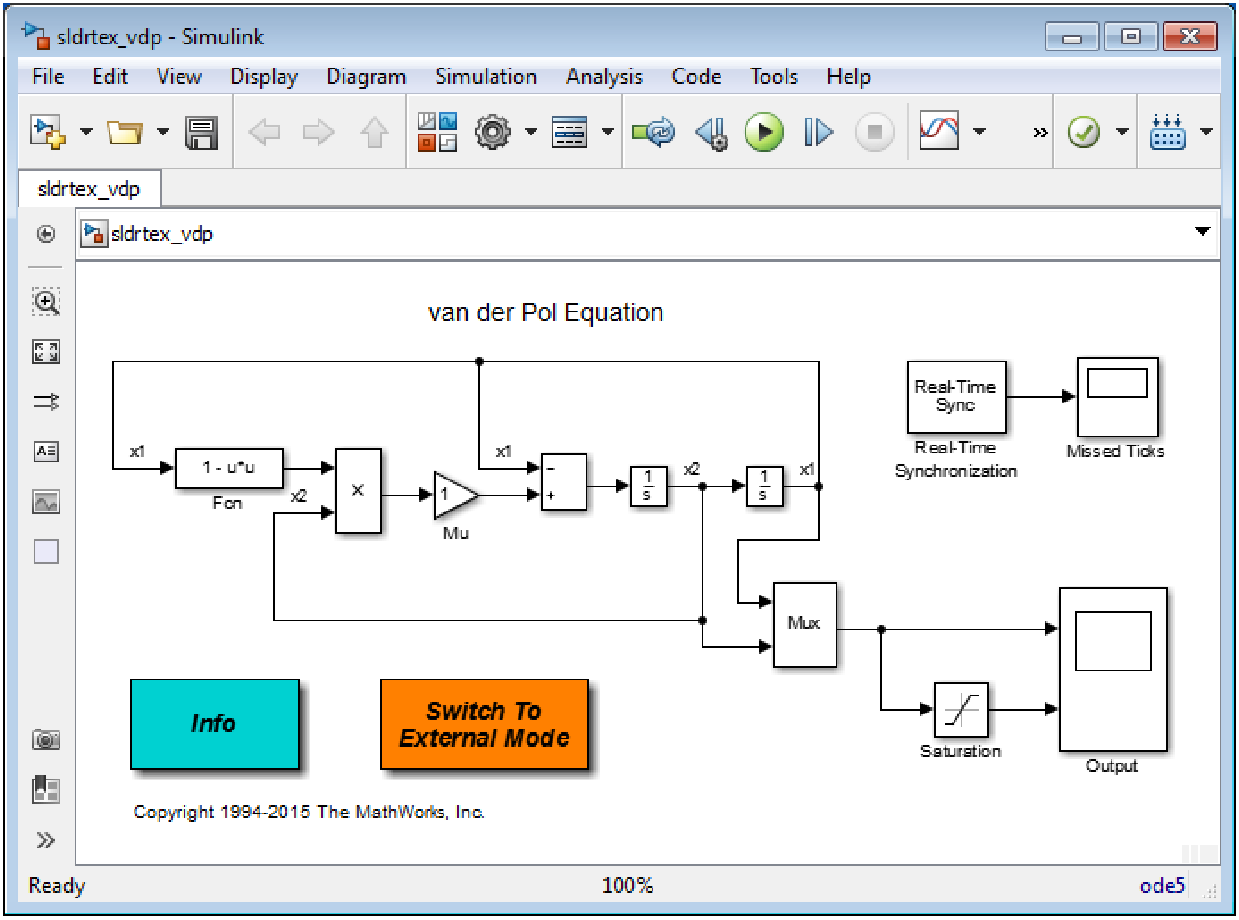
The model `sldrtex_vdp` does not have I/O blocks, so that you can run this model regardless of the I/O boards in your computer. Running this model will test the installation by running Simulink Coder code generation software, Simulink Desktop Real-Time software, and the Simulink Desktop Real-Time kernel.

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 model `sldrtex_vdp` window opens.

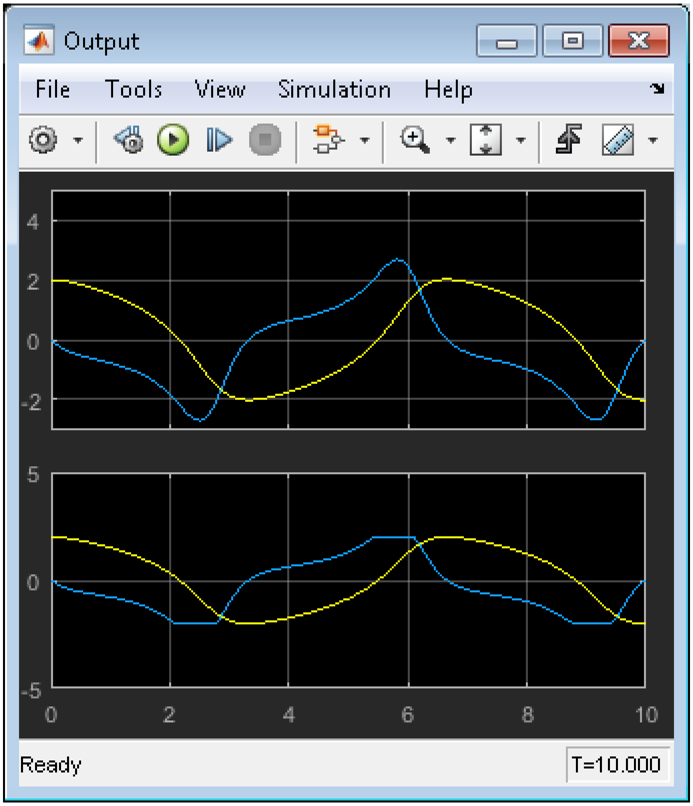


2 Double click **Switch To External Mode** on the model.

Tip The model is initially in normal mode. At this point, you might start real-time normal mode simulation by clicking **Simulation** > **Run**.

3 Click the Run button ▶ on the toolbar.

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



4 To stop the simulation before the end, click the Stop button ■ on the toolbar.

The real-time application stops running, and 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 timer(s) 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 driver(s) are installed on your system at the address and with the parameter(s) 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 Adapter block to match the I/O board installed in your computer.

Note:

- You can run examples in Simulink normal mode (initial setting), accelerator mode, or Simulink external 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.

   The Simulink Desktop Real-Time Demos window displays.
2. From the list, select the example to open it.

Basic Procedures

- “Tune Block Parameters with Data Navigation” on page 3-57
- “Sweep MATLAB Variables with MATLAB Scripting” on page 3-63

Prepare for Real-Time Execution

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.

To prepare your computer:

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.

To prepare your 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. In the Configuration Parameters dialog box, set the parameters in the **Code Generation** pane to support Simulink Desktop Real-Time code generation.
4. Set the Simulink Desktop Real-Time configuration parameters as required by your model.

The next step is to set the simulation mode to normal, accelerator, or external mode to attain the required sample rate, and then run the simulation.

More About

- "Create a Real-Time Application" on page 3-4

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` (`matlab: 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 real-time normal or accelerator mode:

1. “Configure a Model” on page 3-10
2. “Simulate Model in Real-Time Normal Mode” on page 3-16
3. “Simulate Model in Real-Time Accelerator Mode” on page 3-18

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

1. “Set External Mode Code Generation Parameters” on page 3-20
2. “Prepare External Mode Application” on page 3-23
3. “Set External Mode Scope Parameters” on page 3-24
4. “Execute Real-Time Application in External Mode ” on page 3-27

More About

- “Create a Simulink Model” on page 3-5
- “Run Application from MATLAB Command Line” on page 3-29

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 window opens.
3. In the toolbar, open the Simulink 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:

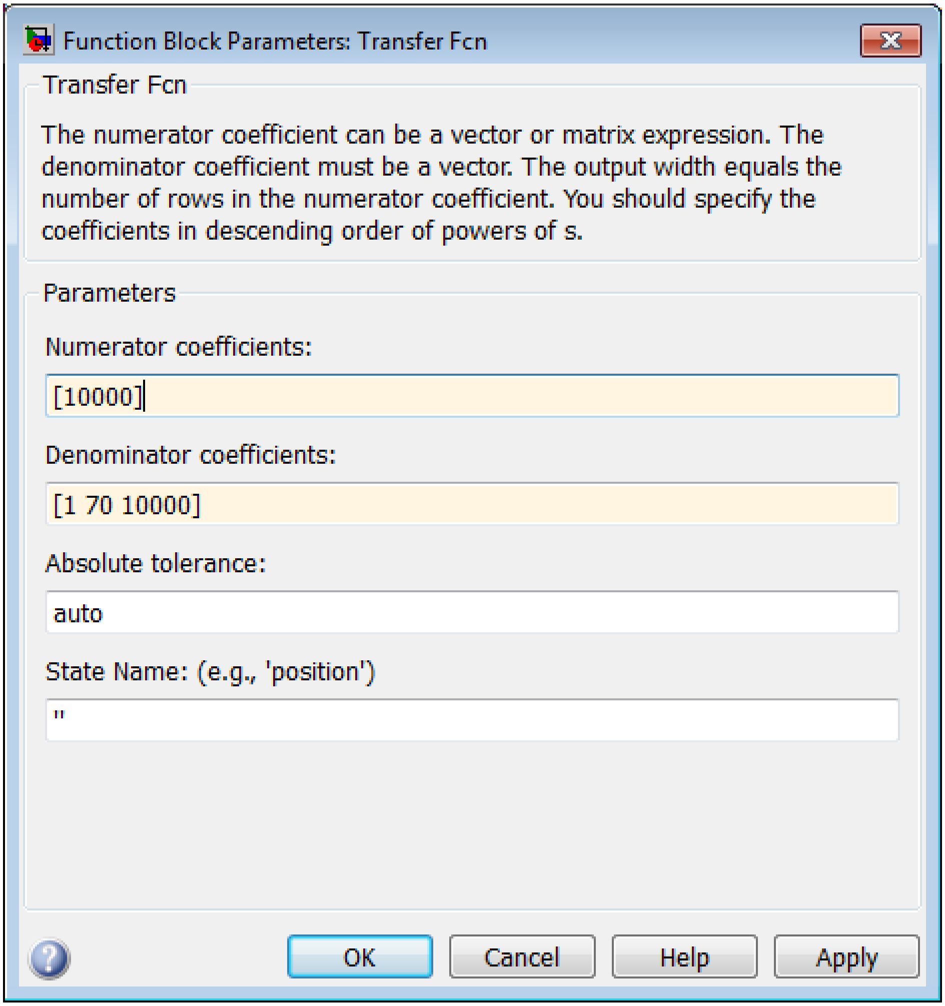
   - Signal Generator output to Transfer Fcn input
   - Transfer Fcn output to Scope input
6. Double-click the Transfer Fcn block. The Block Parameters dialog box opens. In the **Numerator** text box, enter:

   ```
   [10000]
   ```

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

   ```
   [1 70 10000]
   ```

   Click **OK**.



Function 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 70 10000]

Absolute tolerance:

auto

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

''

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

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

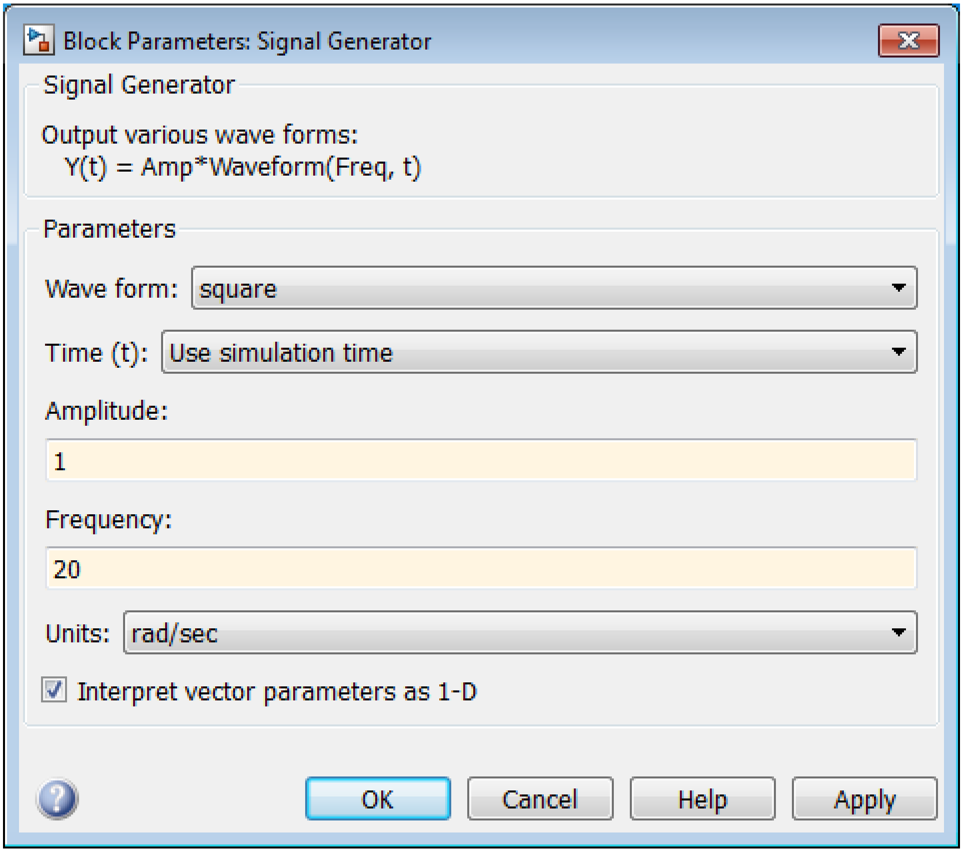
```
1
```

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

20

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

Click **OK**.



Block Parameters: Signal Generator

Signal Generator

Output various wave forms:
Y(t) = Amp*Waveform(Freq, t)

Parameters

Wave form: square

Time (t): Use simulation time

Amplitude:

1

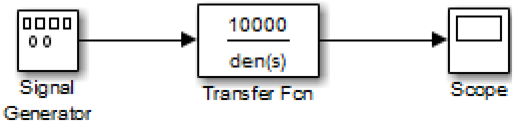
Frequency:

20

Units: rad/sec

☑ Interpret vector parameters as 1-D

The completed Simulink block diagram looks like the figure.



8 From the **File** menu, click **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` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`.

The Simulink Desktop Real-Time software supports model referencing. See "Overview of Model Referencing".

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.

File System I/O

You can use file I/O blocks in Simulink normal or accelerator mode simulation in real time. In these modes, the real-time kernel does not perform I/O, Simulink itself does. To access files in external mode, use the `Packet Input`, `Packet Output`, `Stream Input`, or `Stream Output` blocks and select the driver **Standard Devices > File**.

When run in external mode, the Simulink Desktop Real-Time software does not include a file system. A Simulink model that you intend to use in a Simulink Desktop Real-Time application cannot use blocks, such as the `To File` or `From File` block, that generate file I/O calls such as `fopen` or `fprintf`.

If a Simulink Desktop Real-Time model contains an I/O block, an error can occur when you try to compile or use external 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 3-34. For information about using external mode to execute a Simulink Desktop Real-Time application, see “Execute Real-Time Application in External Mode ” on page 3-27.

Configure a Model

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 a Configuration Set”.

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` (`matlab: 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, from the **File** menu, click **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.

Note: 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.

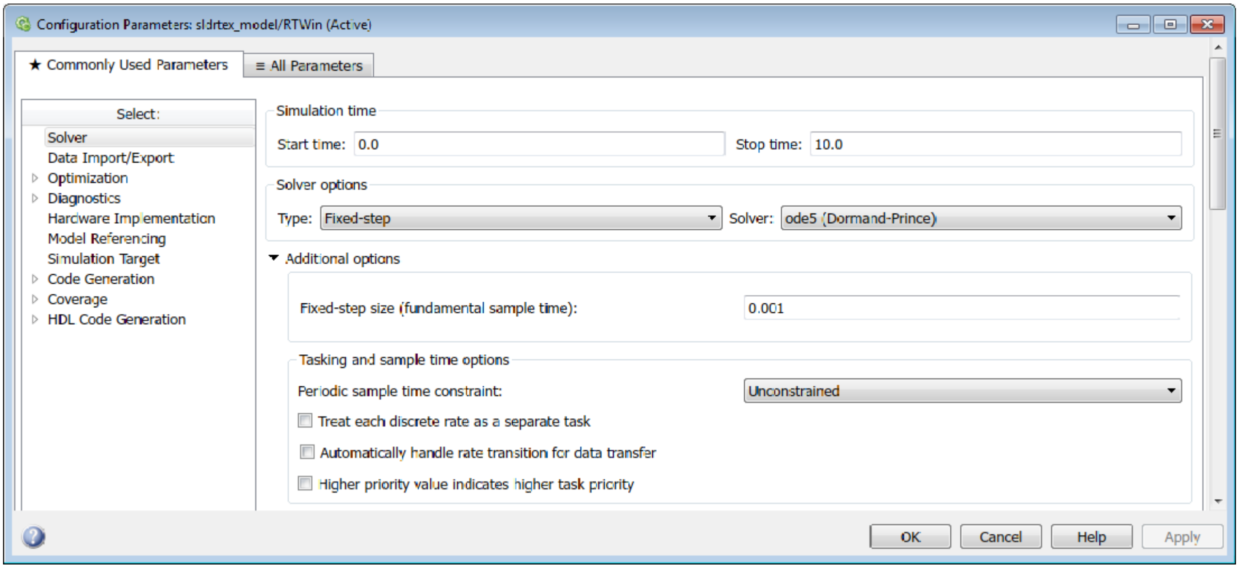
Enter Configuration Parameters Manually

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

This procedure uses the model `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you have already loaded that model.

1. From the **Simulation** menu, click **Model Configuration Parameters**. 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, choose `Fixed-step`. Simulink Coder code generation software does not support variable step solvers.
4. From the **Solver** list, choose a solver. For example, choose 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.)



Simulation time

Start time: 0.0 Stop time: 10.0

Solver options

Type: Fixed-step Solver: ode5 (Dormand-Prince)

Additional options

Fixed-step size (fundamental sample time): 0.001

Tasking and sample time options

Periodic sample time constraint: Unconstrained

□ Treat each discrete rate as a separate task

□ Automatically handle rate transition for data transfer

□ Higher priority value indicates higher task priority

7 Click **OK**.

Enter Scope Parameters for Signal Tracing

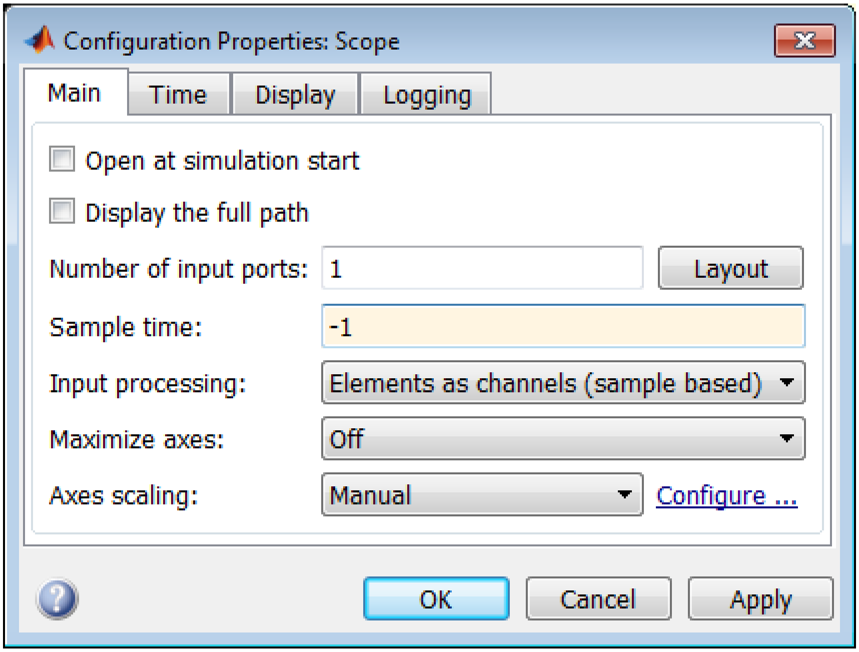
You enter or change scope parameters to specify the *x*-axis and *y*-axis in a Scope window. Other properties include the number of graphs in one Scope window and the sample time for models with discrete blocks.

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

1 In the Simulink window, 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.



Configuration Properties: Scope

□ Open at simulation start

□ Display the full path

Number of input ports: 1

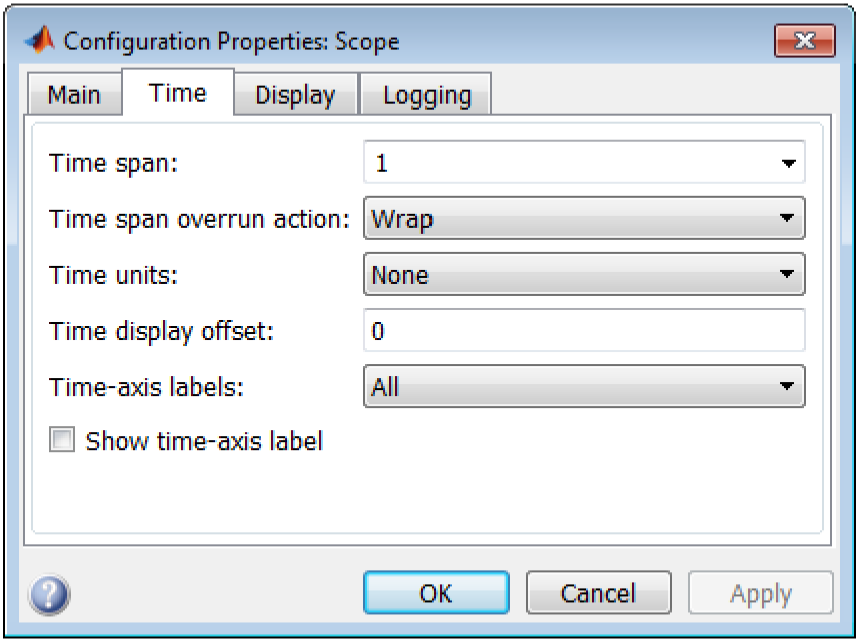
Sample time: -1

Input processing: Elements as channels (sample based)

Maximize axes: Off

Axes scaling: Manual

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



Configuration Properties: Scope

Time span: 1

Time span overrun action: Wrap

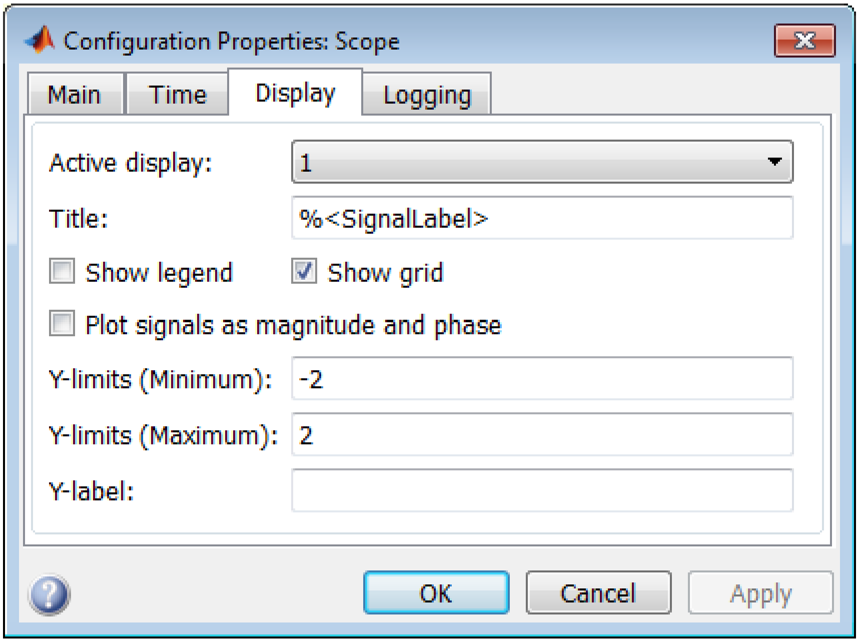
Time units: None

Time display offset: 0

Time-axis labels: All

□ Show time-axis label

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.



Active display: 1

Title: %<SignalLabel>

□ Show legend ☑ Show grid

□ Plot signals as magnitude and phase

Y-limits (Minimum): -2

Y-limits (Maximum): 2

Y-label:

6 Click **OK**.

Simulate Model in Real-Time Normal Mode

You can use Simulink normal mode to run a real-time simulation. This procedure uses the model `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you have loaded that model.

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

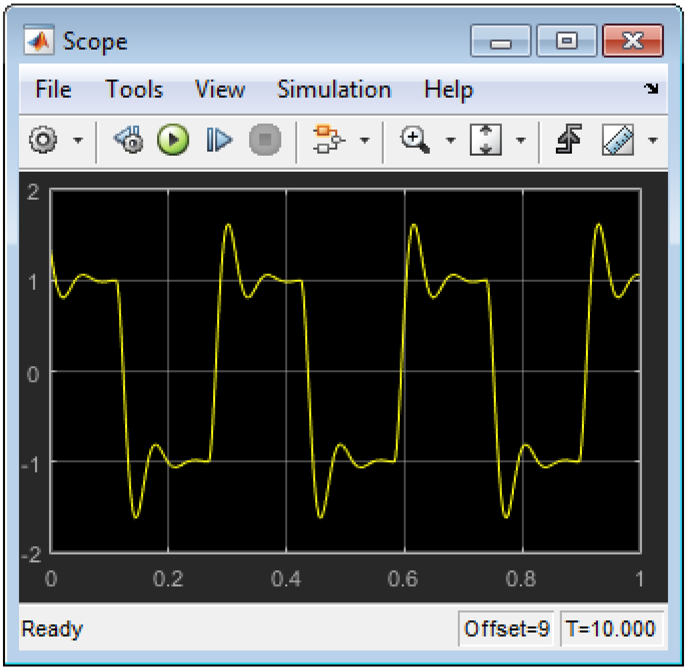
   The Simulink software opens a Scope window with an empty graph.

2 Select **Simulation** > **Mode** > **Normal**.

3 In your Simulink Desktop Real-Time model, set values for the **Sample Time** and **Maximum Missed Ticks** block parameters to prevent missed ticks.

4 To begin simulation, on the toolbar, click the **Run** button ▶.

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



5 To stop the simulation before the end, click the Stop button ■.

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

Simulate Model in Real-Time Accelerator Mode

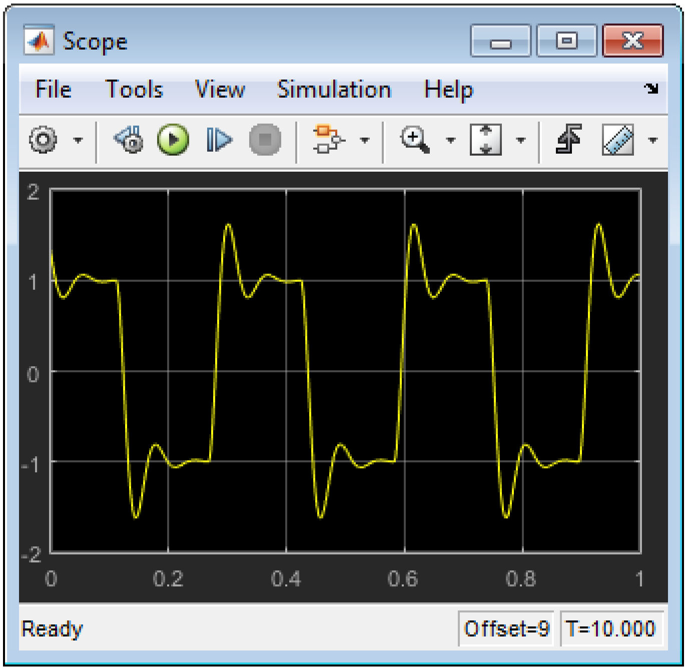
You use Simulink accelerator mode to run a real-time simulation. This procedure uses the model `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`). It assumes that you have loaded that model.

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

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

   The Simulink software opens a Scope window with an empty graph.
2. Select **Simulation** > **Mode** > **Accelerator**.
3. In your Simulink Desktop Real-Time model, set values for the **Sample Time** and **Maximum Missed Ticks** block parameters to prevent missed ticks.
4. To begin simulation, on the toolbar, click the **Run** button ▶.

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



5 To stop the simulation before the end, click the Stop button ■.

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

Set External 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` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you have already loaded that model.

1 In the Simulink window, and from the **Simulation** menu, click **Model Configuration Parameters**.

2 Click the **Code Generation** node.

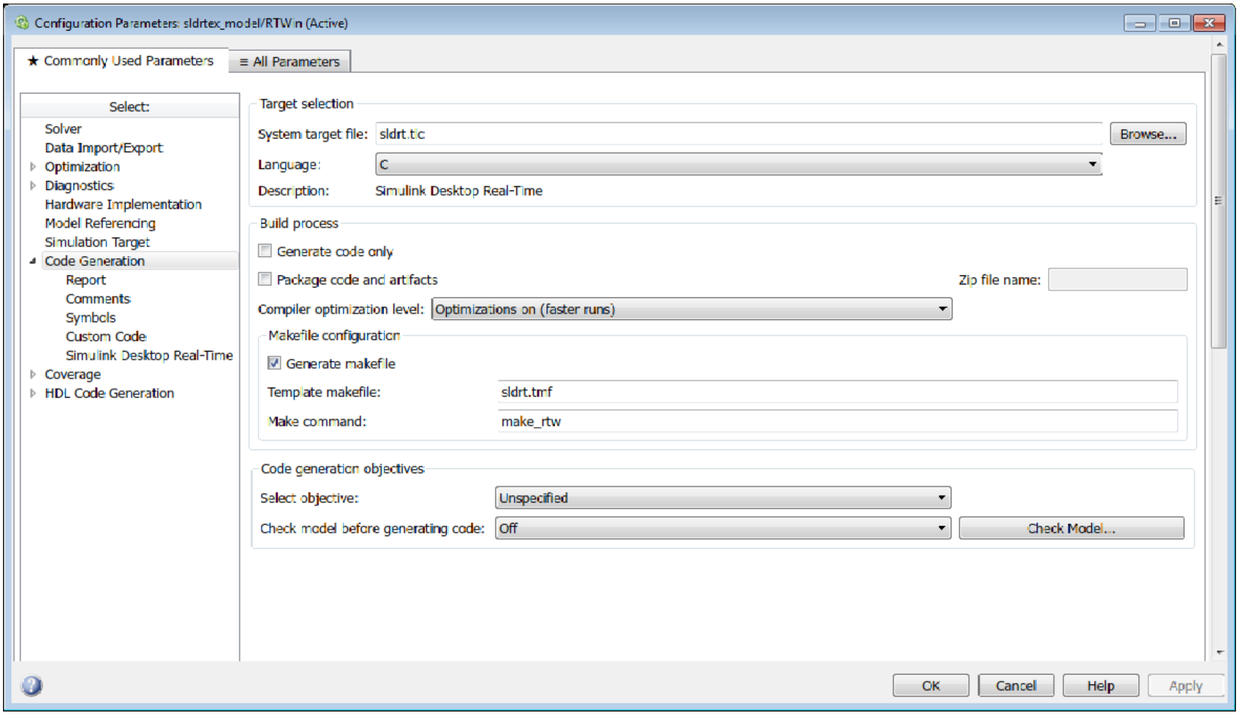
3 In the **Target selection** section, click the **Browse** button at the **System target file** list.

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 external 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.



Target selection

System target file: sldrt.tlc

Language: C

Description: Simulink Desktop Real-Time

Build process

☐ Generate code only

☐ Package code and artifacts   Zip file name:

Compiler optimization level: Optimizations on (faster runs)

Makefile configuration

☑ Generate makefile

Template makefile: sldrt.tmf

Make command: make_rtw

Code generation objectives

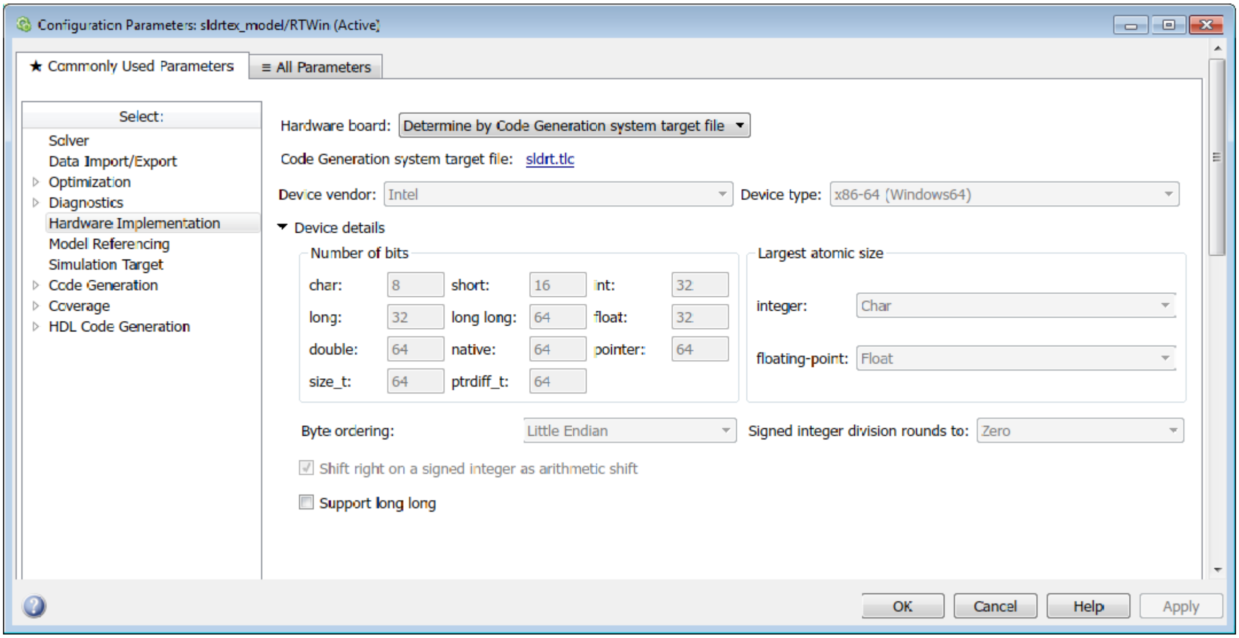
Select objective: Unspecified

Check model before generating code: Off

Do not set **Default parameter behavior** to `Inlined` on the **Signals and Parameters** node under **Optimization**. 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 host computer. For example, for a 64-bit Intel® machine, they are:

- **Device vendor** — `Intel`
- **Device type** — `x86-64`



Configuration Parameters: sldrtex_model/RTWin (Active)

Hardware board: Determine by Code Generation system target file

Code Generation system target file: sldrt.tlc

Device vendor: Intel

Device type: x86-64 (Windows64)

Device details

Number of bits

char:	8	short:	16	int:	32
long:	32	long long:	64	float:	32
double:	64	native:	64	pointer:	64
size_t:	64	ptrdiff_t:	64		

Largest atomic size

integer: Char

floating-point: Float

Byte ordering: Little Endian

Signed integer division rounds to: Zero

☑ Shift right on a signed integer as arithmetic shift

□ Support long long

6 Click **OK**.

Prepare External Mode Application

In external 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` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you have already loaded that model.

1. In the Configuration Parameters dialog box, enter parameters for use by the Simulink Coder code generation software. See “Set External Mode Code Generation Parameters” on page 3-20.
2. On the toolbar, click the **Build** .

- 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 `sldrtex_model.rwd` by using `sldrtex_model.mk`. The file `sldrtex_model.rwd` is a binary file that is referred to as your 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, start MATLAB again, and then connect and run the executable without having to rebuild your code.

Set External Mode Scope Parameters

Simulink normal 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.

Simulink external 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 normal or accelerator mode.

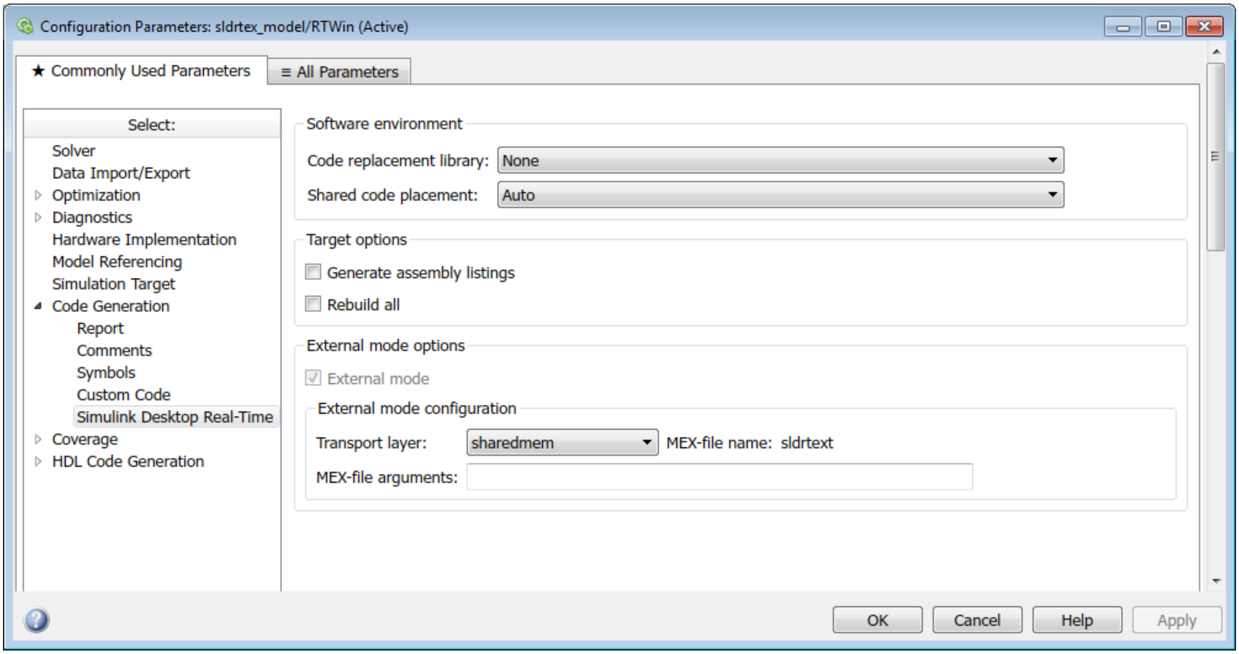
This procedure uses the model `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you have already loaded that model.

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

1. In the Simulation window, from the **Simulation** menu, click **Model Configuration Parameters**.
2. 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.

   Verify 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 external mode and supports uploading signal data and downloading parameter values.

   Click **OK**.



Configuration Parameters: sldrtex_model/RTWin (Active)

Software environment

Code replacement library: None

Shared code placement: Auto

Target options

□ Generate assembly listings

□ Rebuild all

External mode options

☑ External mode

External mode configuration

Transport layer: sharedmem MEX-file name: sldrtext

MEX-file arguments:

4 In the Simulation window, click **Code** > **External Mode Control Panel**.

Click the **Signal & Triggering** button.

5 Select the **Select all** check box. From the **Source** list, choose `manual`. From the **Mode** list, select `normal`.

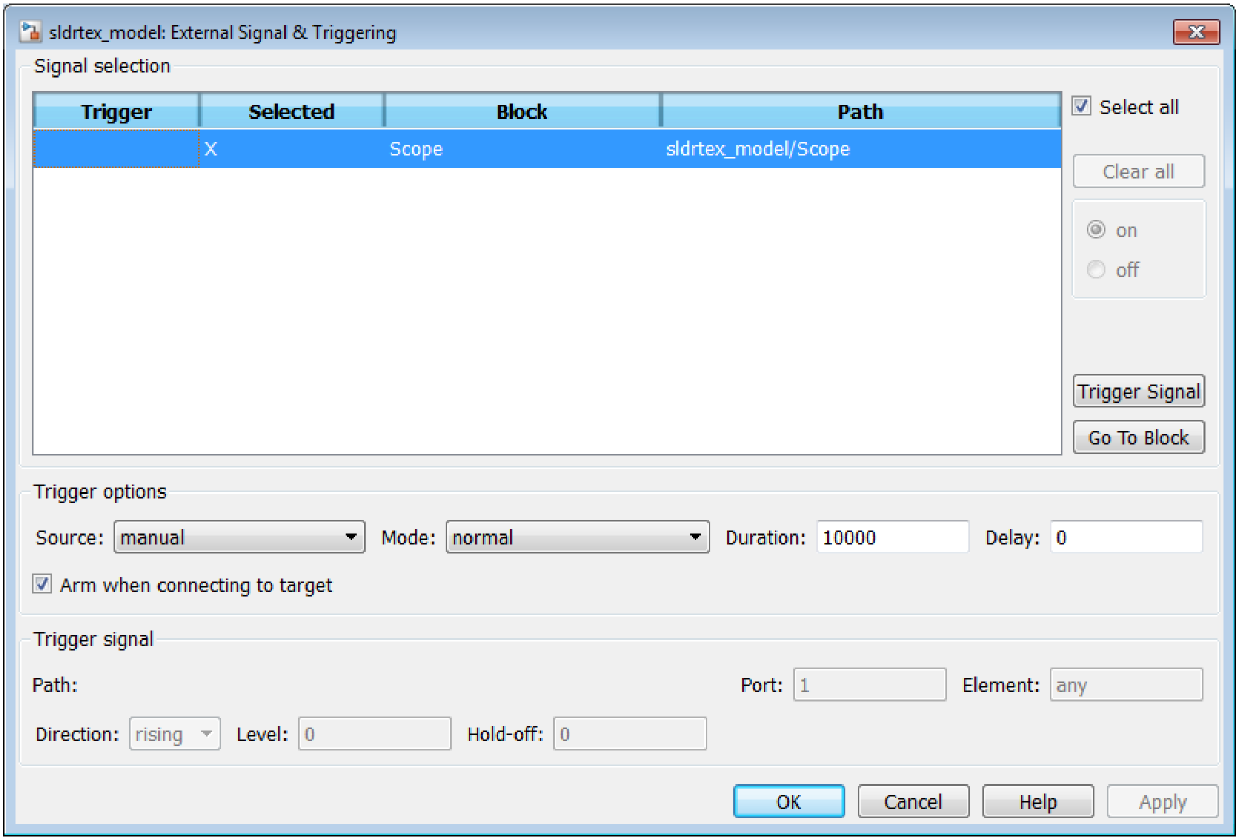
The `X` under **Signal selection** indicates that a signal is tagged for data collection. `T` indicates that the signal is tagged as a trigger signal.

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.



sldrtex_model: External Signal & Triggering

Signal selection

Trigger	Selected	Block	Path
	X	Scope	sldrtex_model/Scope

☑ Select all

● on
○ off

Trigger options

Source: manual

Mode: normal

Duration: 10000

Delay: 0

☑ Arm when connecting to target

Trigger signal

Path:

Port: 1

Element: any

Direction: rising

Level: 0

Hold-off: 0

8 Click **Close**.

Execute Real-Time Application in External Mode

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

This procedure uses the model `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you have already created a real-time application from that model.

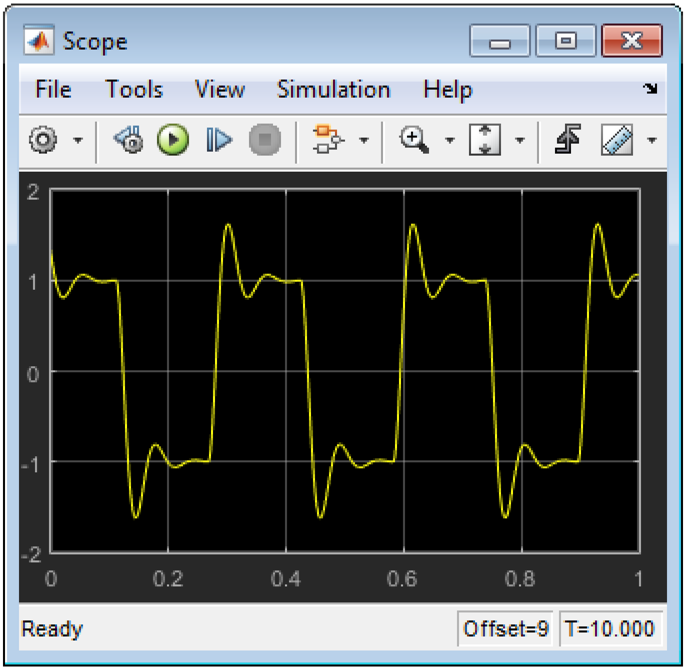
In external mode, you execute your real-time application to observe the behavior of your model in real time with the generated code. You cannot run a real-time application in **Rapid Accelerator** mode.

- To allow exchange of commands, parameters, and logged data, establish a connection between your Simulink model and the kernel.
- Run the application in real time.

1. To set external mode, select **Simulation** > **Mode** > **External**.
2. To start real-time application execution, on the toolbar, click the **Run** button ▶.
3. To stop the simulation before the end, click the **Stop** button ■.

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.


Scope
File
Tools
View
Simulation
Help
2
1
0
-1
-2
0
0.2
0.4
0.6
0.8
1
Ready
Offset=9
T=10.000

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` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`). It assumes that you have already created a real-time application from that model.

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')
```

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')
```

Inspect Simulink Desktop Real-Time Signals with Simulation Data Inspector

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

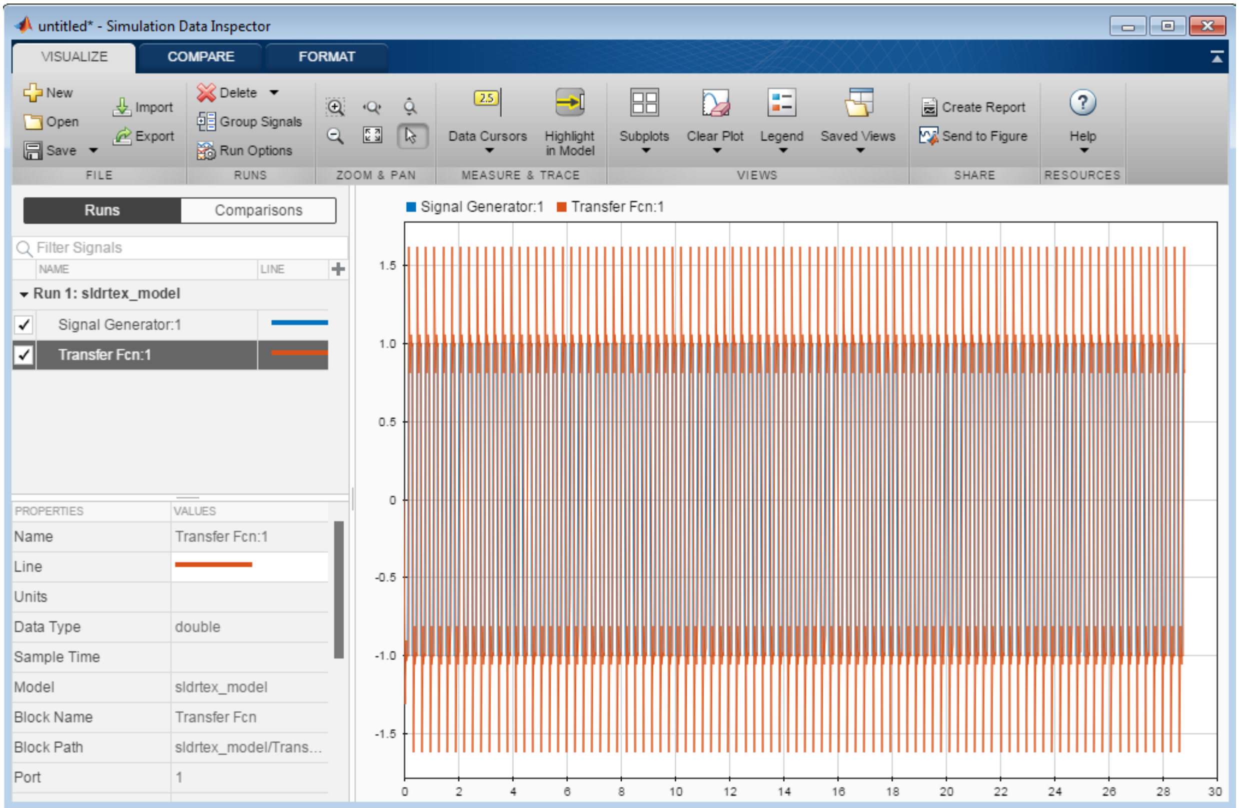
This procedure uses the model `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. Make sure that you have started the Simulink Desktop Real-Time kernel.

1. Open `sldrtex_model`.
2. On the toolbar, set the simulation stop time to, for example, `30` seconds.
3. Click **Simulation** > **Mode** > **External**.
4. In the model, select the signals `Signal Generator` and `Transfer Fcn`.
5. On the toolbar, click the arrow next to the **Simulation Data Inspector** button, and then select **Stream Selected Signals to Data Inspector**.

   A faint **Simulation Data Inspector** icon appears next to each signal.
6. To start the simulation, click the **Run** button.

   The **Simulation Data Inspector** button glows, indicating that Simulation Data Inspector has data available for viewing.
7. Click the glowing **Simulation Data Inspector** button.
8. In Simulation Data Inspector, select the signals `Signal Generator:1` and `Transfer Fcn:1`.

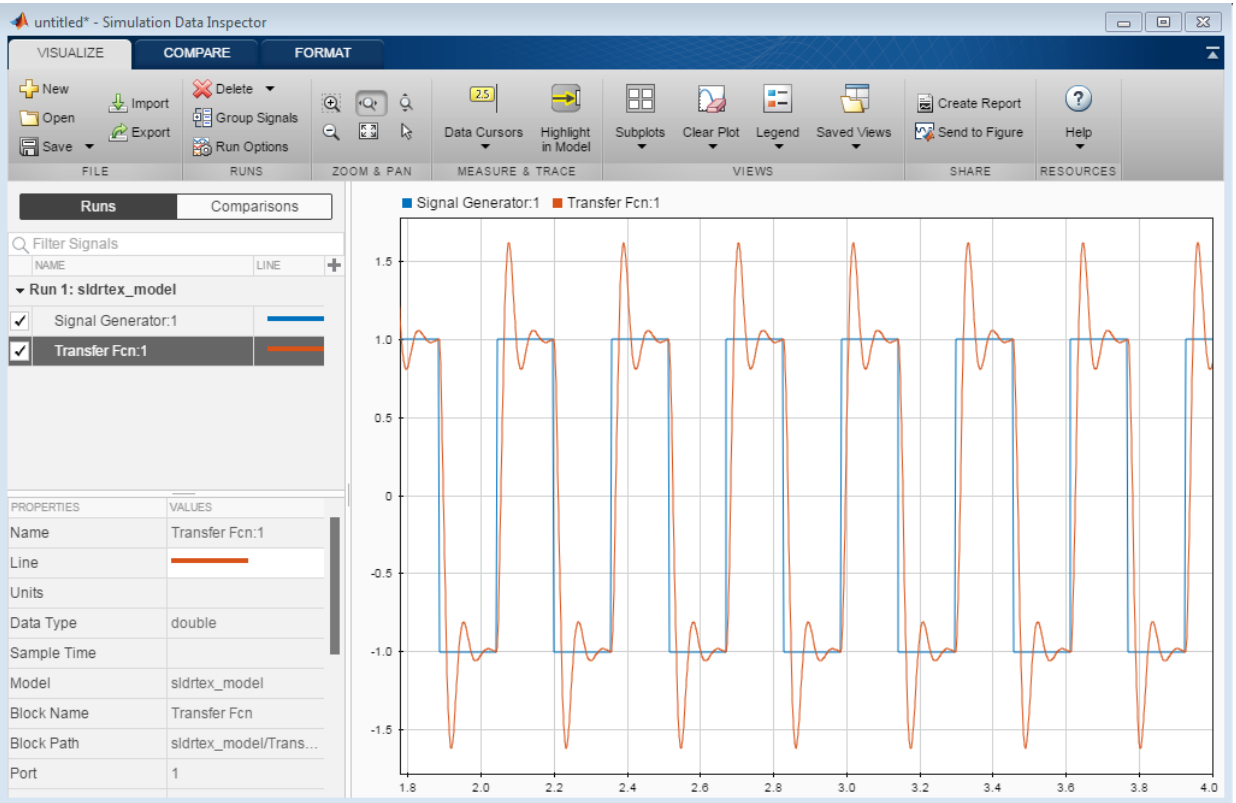
   Simulation Data Inspector displays plotted data.



PROPERTIES	VALUES
Name	Transfer Fcn:1
Line	🟥
Units	
Data Type	double
Sample Time	
Model	sldrtex_model
Block Name	Transfer Fcn
Block Path	sldrtex_model/Trans...
Port	1

9 To stop the simulation, click the **Stop** button ■.

10 After the simulation, you can 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.



PROPERTIES	VALUES
Name	Transfer Fcn:1
Line	🟥
Units	
Data Type	double
Sample Time	
Model	sldrtex_model
Block Name	Transfer Fcn
Block Path	sldrtex_model/Trans...
Port	1

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

More About

- “Water Tank Model with Dashboard”
- “Inspect Signal Data with Simulation Data Inspector”

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.

If your model contains `Outport` blocks, you cannot save signal data in Simulink external mode. Simulink supports signal logging with `Outport` blocks in normal or accelerator modes only, when Simulink runs the simulation algorithm.

Tip In external 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.

Set Scope Parameters for Logging to Workspace

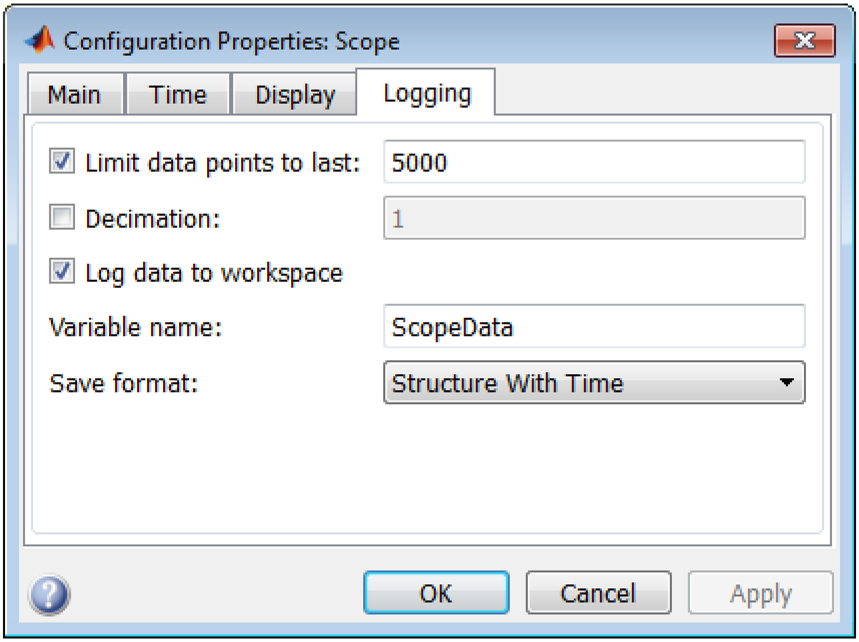
Data is saved to the MATLAB workspace through a Simulink Scope block. For data to be saved, set Scope block parameters. After you create a Simulink model and add a Scope block, you can enter the scope parameters for signal logging to the MATLAB workspace.

This procedure uses the model `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model')))`). It assumes that you have already loaded that model.

1. In the Simulink window, double-click the `Scope` block.
2. In the Scope window, on the toolbar, click the **Parameters** button ⚙.
3. In the Scope Parameters dialog box, click the **Logging** tab.
4. Do one of the following:

   - If you are running a simulation, select the **Limit data points to last** check box, and enter the number of sample points to save.
   - If you are running an execution, 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 External Mode Properties for Logging to Workspace” on page 3-37. For more information, see “Set Up an External Mode Communication Channel”.
5. 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`.
6. From the **Save format** list, choose among `Structure with time`, `Structure`, `Array`, and `Dataset`. For example, to save the sample times and signal values at those times, choose `Structure with time`.



☑ Limit data points to last: 5000

□ Decimation: 1

☑ Log data to workspace

Variable name: ScopeData

Save format: Structure With Time

7 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.

Set External 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 normal 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 `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you have already loaded that model.

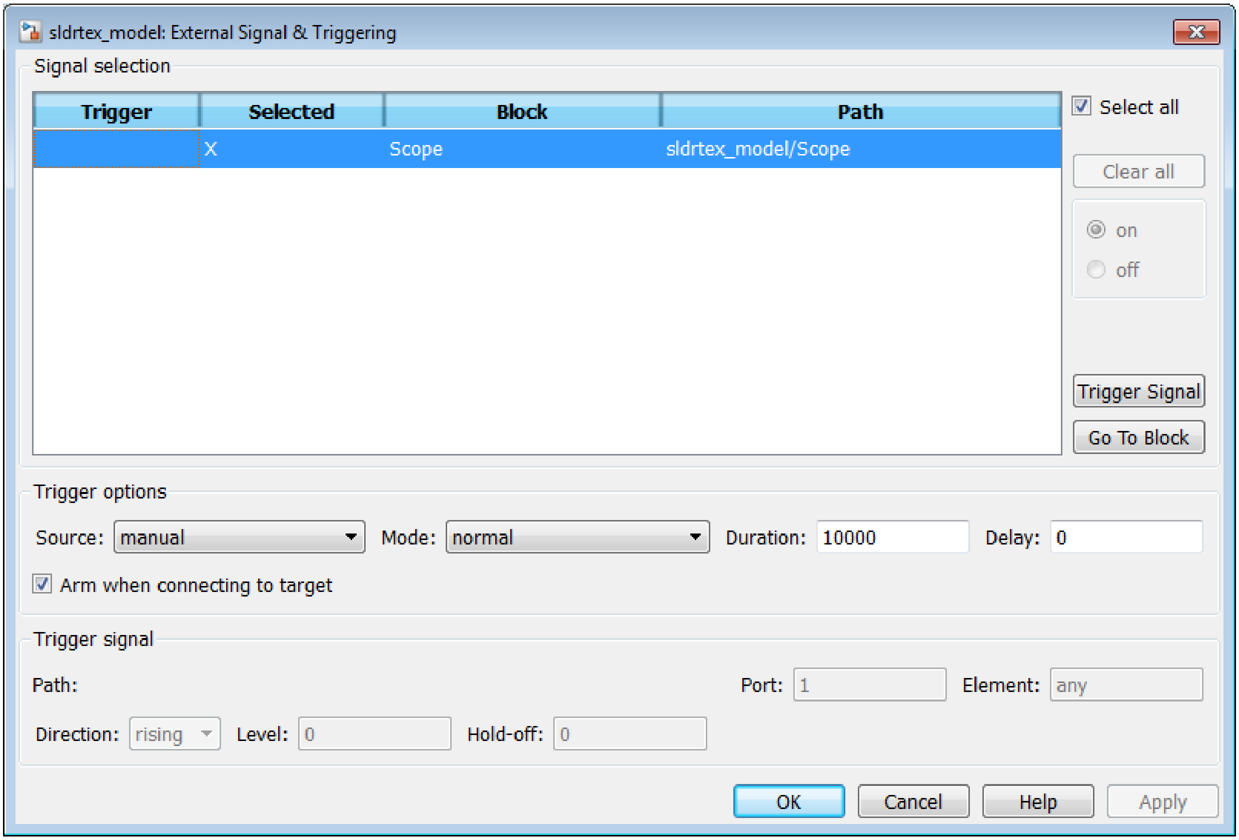
1. In the Simulation window, click **Code** > **External Mode Control Panel**.
2. Click the **Signal & Triggering** button.
3. Click the **Select all** button. From the `Source` list, choose `manual`. From the `Mode` list, choose `normal`.

   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:

   ```
   10000
   ```

   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 3-35.

   For more information, see “Set Up an External Mode Communication Channel”.



sldrtex_model: External Signal & Triggering

Signal selection

Trigger	Selected	Block	Path
	X	Scope	sldrtex_model/Scope

☑ Select all

Trigger options

Source: manual Mode: normal Duration: 10000 Delay: 0

☑ Arm when connecting to target

Trigger signal

Path: Port: 1 Element: any

Direction: rising Level: 0 Hold-off: 0

6 Click **Close**.

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

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 `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you saved your data to the variable `ScopeData`.

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

```
ScopeData
```

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

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

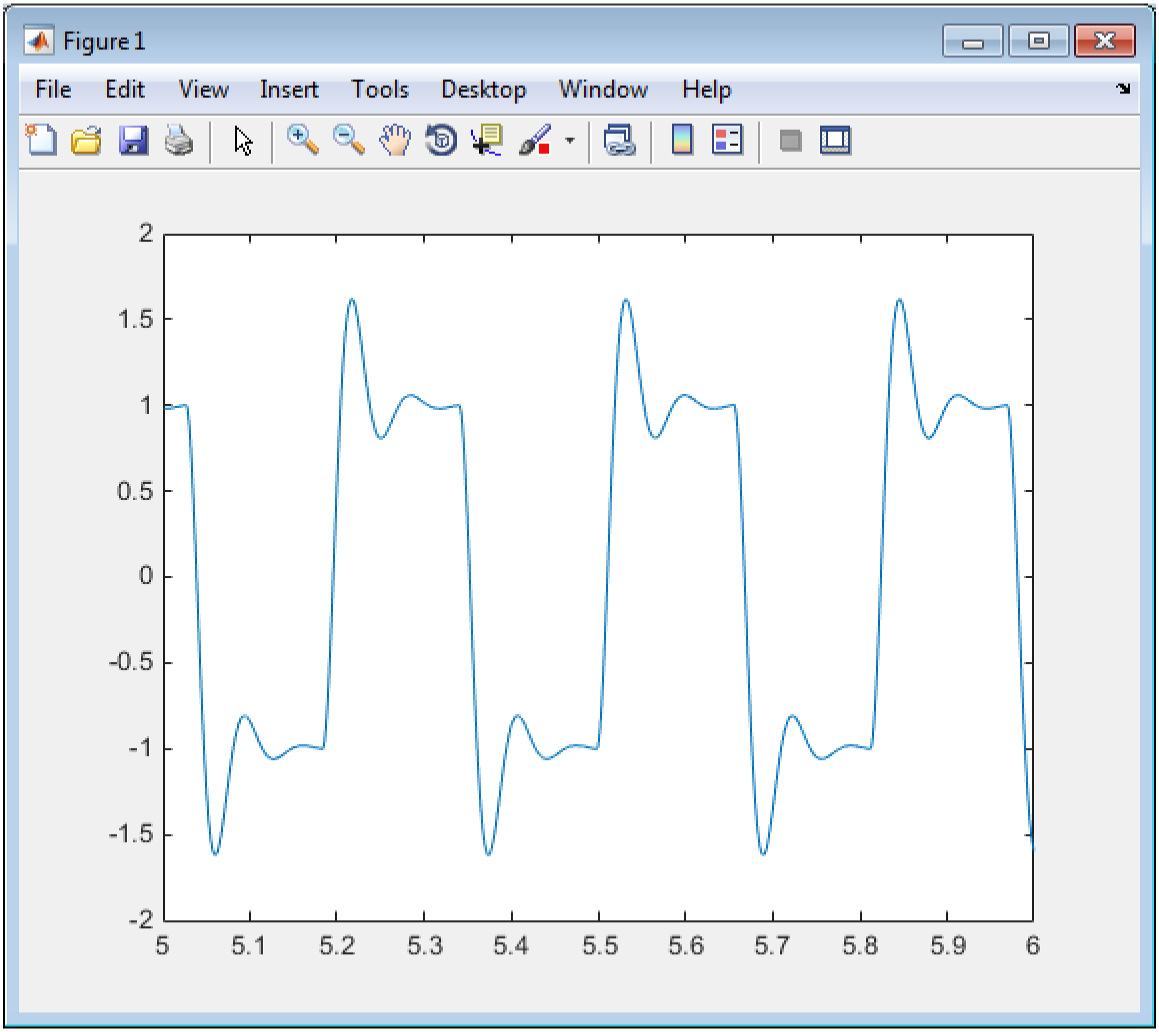
```
ScopeData.signals
```

```
ans =
       values: [10000x1 double]
   dimensions: 1
        label: ''
        title: []
    plotStyle: 1
```

2 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.



3 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`.

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.

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

Tip In external 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.

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.

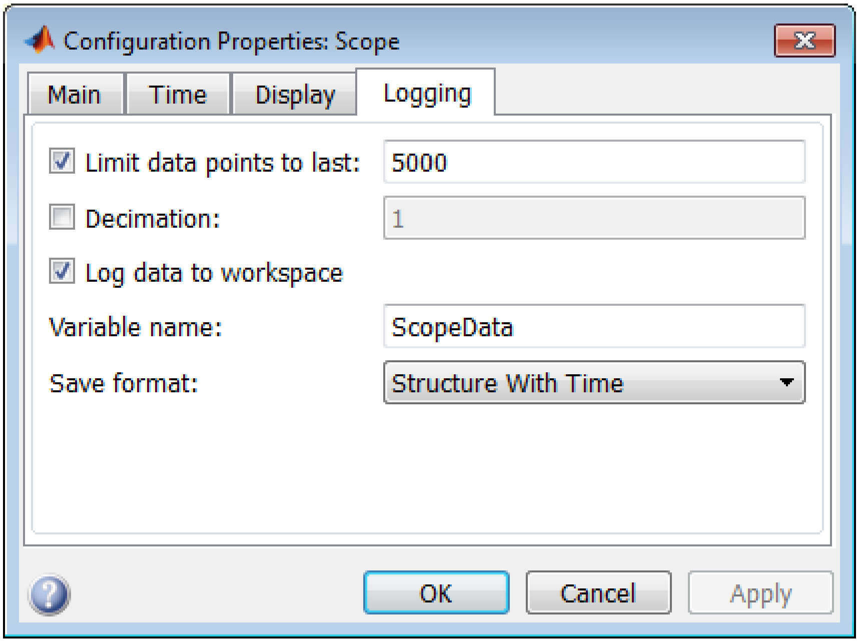
This procedure uses the model `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`). It assumes that you have already loaded that model.

1. In the Simulink window, double-click the Scope block.
2. On the toolbar, click the **Parameters** button ⚙.
3. Click the **Logging** tab.
4. Do one of the following:

   - If you are running a simulation, select the **Limit data points to last** check box, and enter the number of sample points to save.
   - If you are running an execution, 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 External Mode Properties for Logging to File" on page 3-44. For more information, see "Set Up an External Mode Communication Channel".
5. 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 **Save data to workspace** check box to be able to save data to a file. If you do not select the **Save data to workspace** check box, the MAT-files for data logging are created, but they are empty.
6. From the **Save format** list, choose among `Structure with time`, `Structure`, `Array`, and `Dataset`. For example, to save the sample times and signal values at those times, choose `Structure with time`.



☑ Limit data points to last: 5000

□ Decimation: 1

☑ Log data to workspace

Variable name: ScopeData

Save format: Structure With Time

7 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.

Set External 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 `sldrtex_model` (`matlab:open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you have already loaded that model.

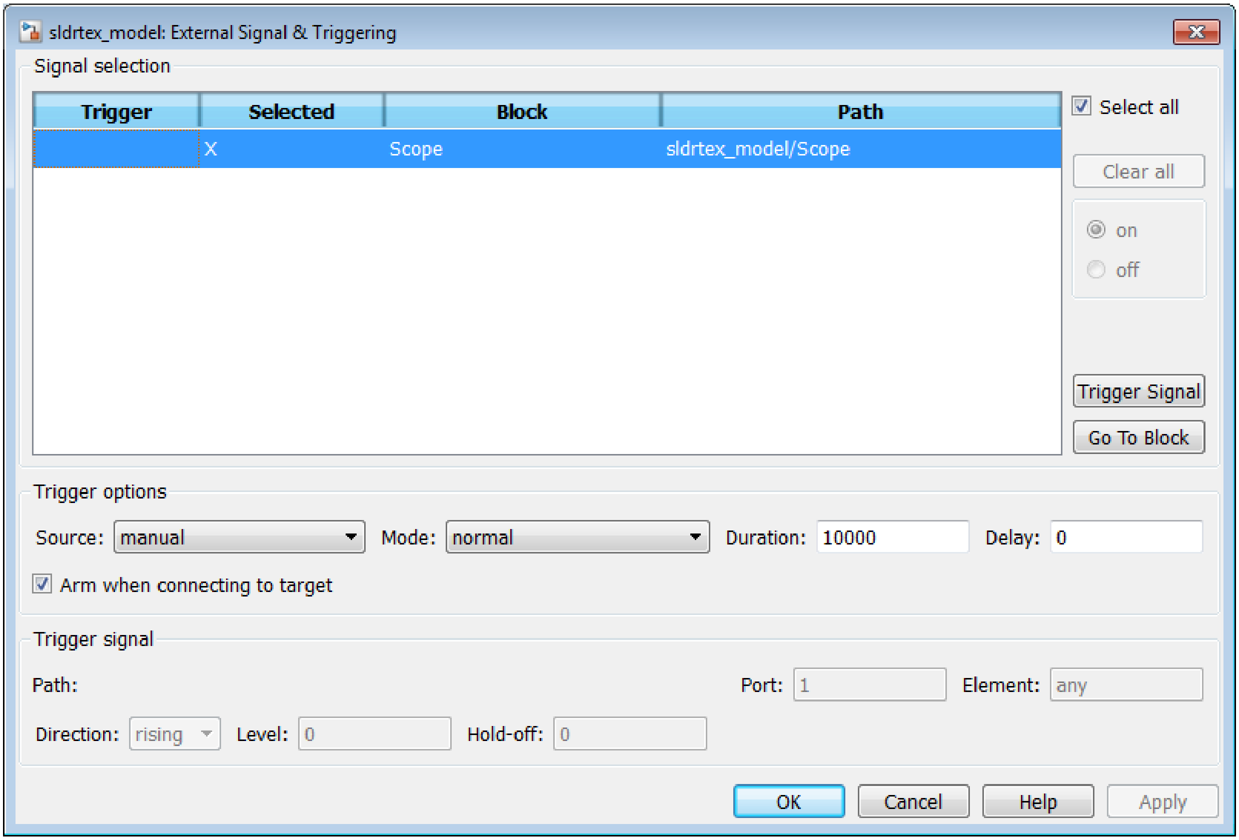
1. In the Simulation window, click **Code** > **External Mode Control Panel**.
2. Click the **Signal & Triggering** button.
3. Click the **Select all** button. From the `Source` list, choose `manual`. From the `Mode` list, choose `normal`.

   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:

   ```
   10000
   ```

   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 3-35.

   For more information, see “Set Up an External Mode Communication Channel”.



sldrtex_model: External Signal & Triggering

Signal selection

Trigger	Selected	Block	Path
	X	Scope	sldrtex_model/Scope

☑ Select all

● on
○ off

Trigger options

Source: manual

Mode: normal

Duration: 10000

Delay: 0

☑ Arm when connecting to target

Trigger signal

Path:

Port: 1

Element: any

Direction: rising

Level: 0

Hold-off: 0

6 Click **Close**.

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

Set External Mode Data Archiving Parameters

You must select the **Save 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` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`). It assumes that you have already loaded that model.

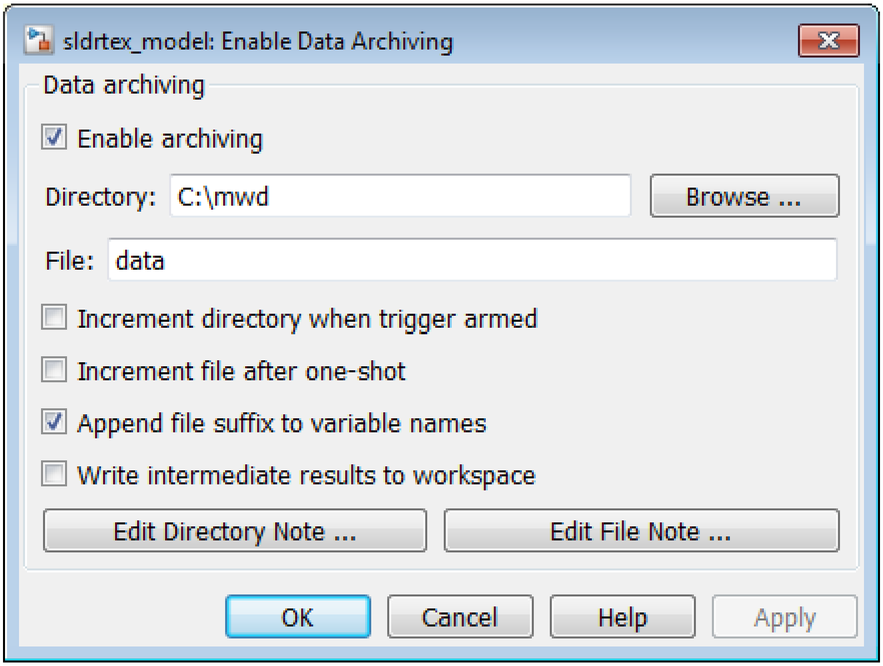
After you create a Simulink model, you can enter the Data Archiving Parameters for data logging to a file:

1. In the Simulation window, click **Code** > **External Mode Control Panel**.
2. 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.



sldrtex_model: Enable Data Archiving

Data archiving

☑ Enable archiving

Directory: C:\mwd

File: data

□ Increment directory when trigger armed

□ Increment file after one-shot

☑ Append file suffix to variable names

□ Write intermediate results to workspace

7 Click the **Close** button.

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 `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. It assumes that you saved your data to the variable `ScopeData`.

1 In the MATLAB Command Window, type:

```
ScopeData
```

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

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

```
dir *.mat
```

```
ScopeData.mat
```

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

```
clear
load ScopeData
who
```

```
Your variables are:
ScopeData
```

4 To plot the first 1000 points, type:

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

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

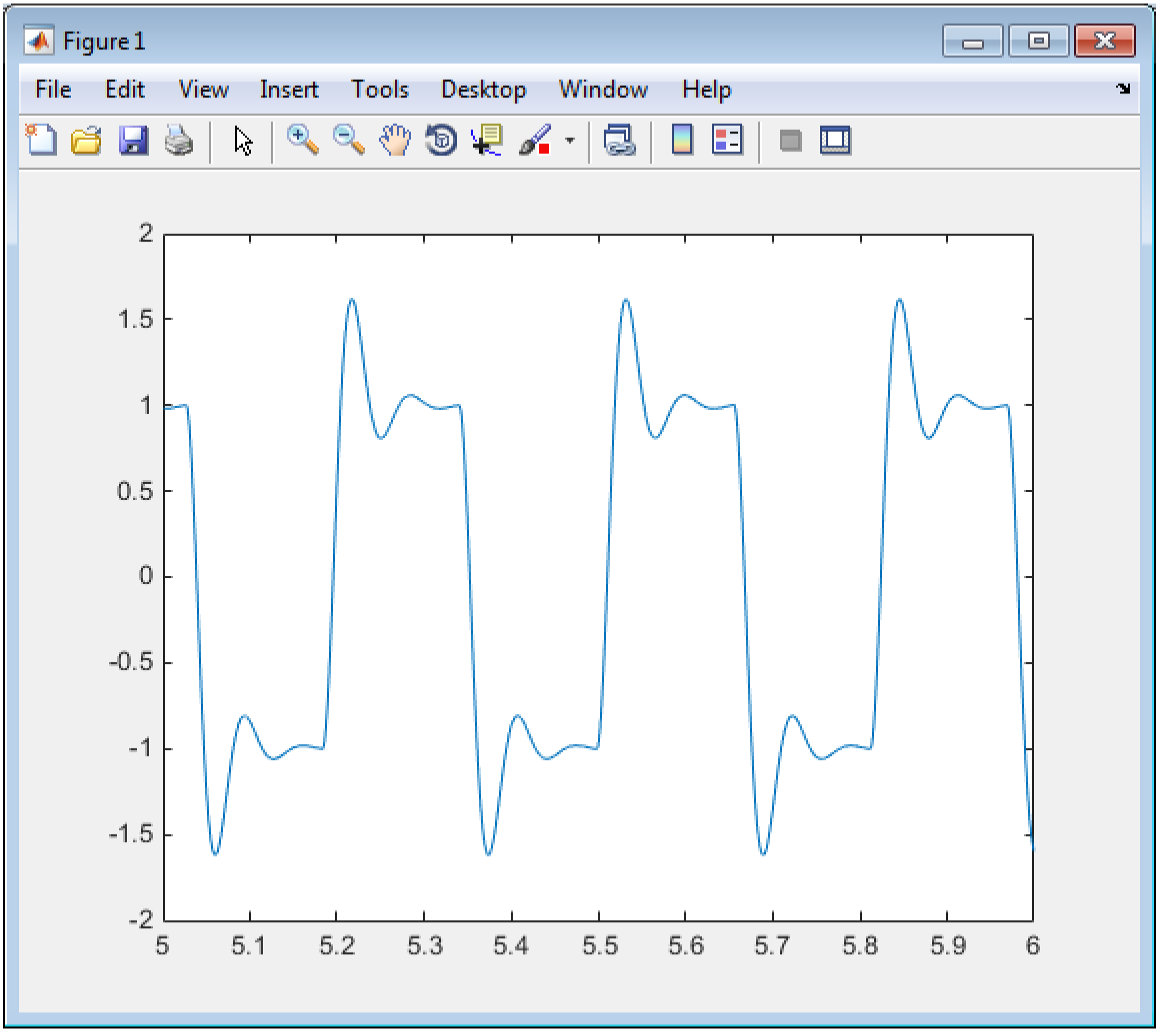


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
5
5.1
5.2
5.3
5.4
5.5
5.6
5.7
5.8
5.9
6

Tunable Block Parameters and MATLAB Variables

To change the behavior of a model, you can tune Simulink Desktop Real-Time block parameters, provided the parameters are tunable. You can change the parameters directly in the block dialog boxes or indirectly by using MATLAB variables.

In normal or accelerator mode, Simulink transfers the new values to the model that is being simulated. In external mode, Simulink transfers the new values to the real-time application that is running in the kernel mode process.

Tunable Parameters

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

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` in the **Signals and Parameters** pane of the **Optimization** node. 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`.

MATLAB Variables

A tunable MATLAB variable is a variable that you reference in a Simulink block dialog box. You can tune a variable 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 MATLAB variable 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, you can set the **Default parameter behavior** option to `Inlined` in the **Signals and Parameters** pane of the **Optimization** node.

By default, you cannot tune inlined block parameters. However, you can create a MATLAB variable or `Simulink.Parameter` object and reference it 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”.

Tuning with External Mode

In external 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 variable, command Simulink to transfer the data by pressing **Ctrl+D** or clicking **Simulation** > **Update Diagram**.

Tuning with MATLAB Language

In Simulink Desktop Real-Time, you can use the MATLAB language command `set_param` to change the values of block parameters and MATLAB variables. The following code examples use the model `sldrtex_model` (`matlab: 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 MATLAB variable, 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')
```

More About

- “Tune Block Parameters with Block Dialog Box” on page 3-53
- “Tune Block Parameters with Data Navigation” on page 3-57
- “Sweep MATLAB Variables with MATLAB Scripting” on page 3-63
- “Water Tank Model with Dashboard”
- “Default parameter behavior”
- “Tune and Experiment with Block Parameter Values”
- “Share and Reuse Block Parameter Values by Creating Variables”
- “Block Parameter Representation in the Generated Code”
- “Configure Block Parameter Tunability for Rapid Prototyping”

Tune Block Parameters with Block Dialog Box

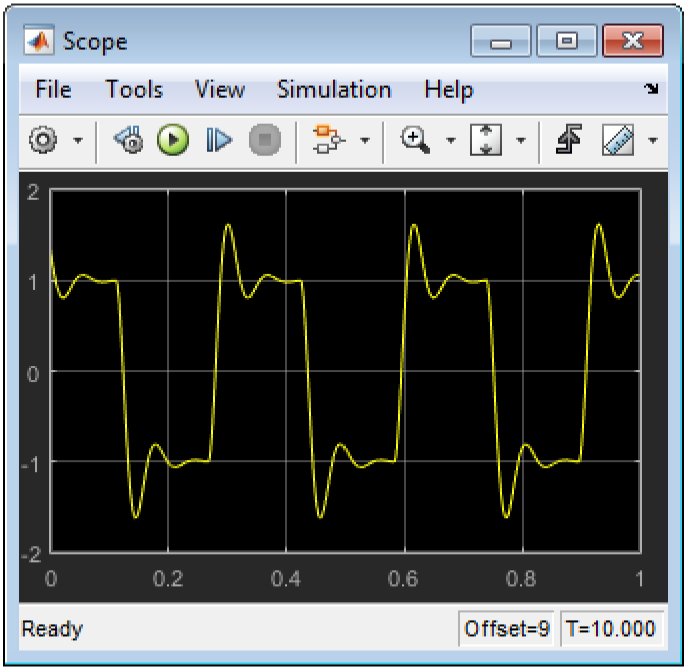
After running your real-time application, use the block mask dialog box or Property Inspector to change parameter values and observe the changes to the signals. In normal or accelerator mode, Simulink transfers the new values to the model that is being simulated. In external mode, Simulink transfers 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` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model'))))`. This model opens in normal mode.

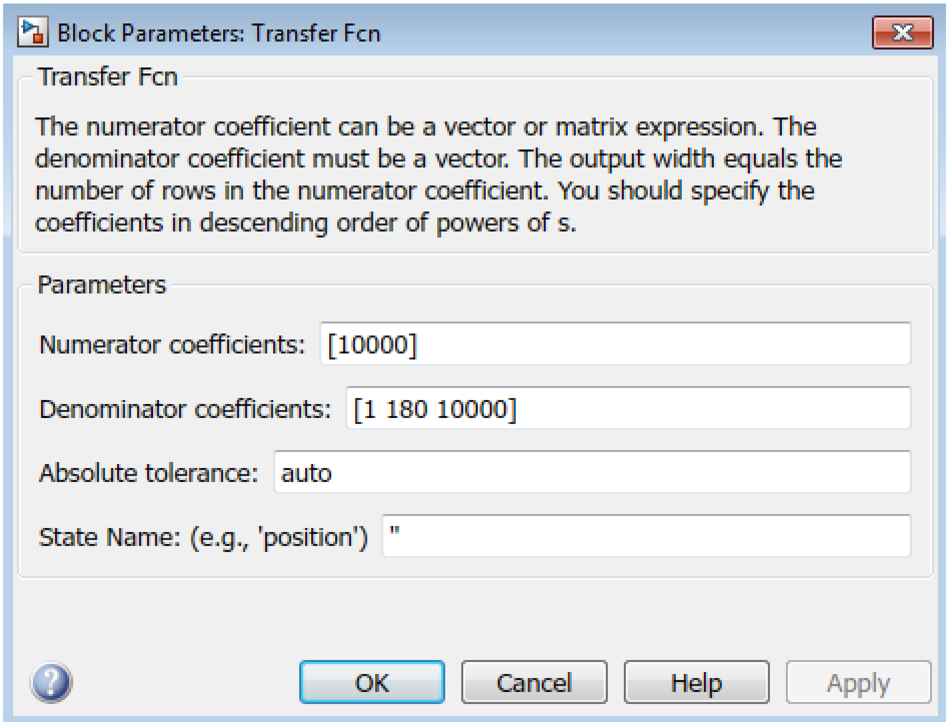
First, install the Simulink Desktop Real-Time kernel and `cd` to a working folder.

1. Open `sldrtex_model`.
2. Open the Scope block.
3. In Simulink Editor, change **Simulation stop time** to `Inf`.
4. Set **Simulation** > **Mode** to `External`.
5. To start execution, on the toolbar, click the **Run** button ▶.



6 Open the Transfer Fcn block parameters dialog box.

7 Change **Denominator coefficients** to `[1 180 10000]`.



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

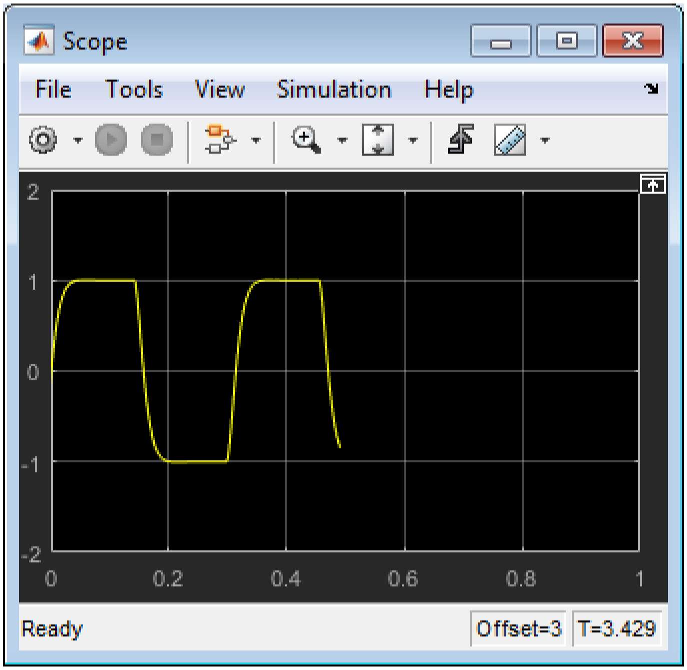
Numerator coefficients: [10000]

Denominator coefficients: [1 180 10000]

Absolute tolerance: auto

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

8 Click **Apply**.



9 Click **Stop**.

More About

- “Tunable Block Parameters and MATLAB Variables” on page 3-50

Tune Block Parameters with Data Navigation

You can embed MATLAB variables in block dialog boxes by using data navigation and then change the variable values during execution. In normal or accelerator mode, Simulink transfers the new values to the model that is being simulated. In external 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.

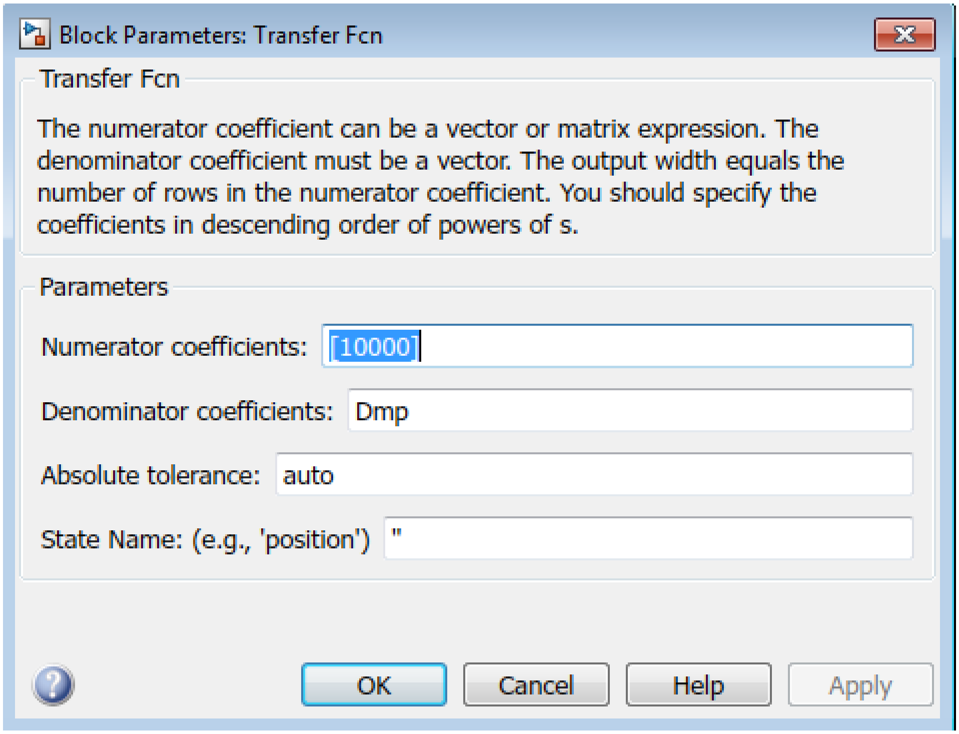
For this example, your goal is to minimize ringing in the transfer function.

This procedure begins with the square-wave transfer function model `sldrtex_model` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_model')))`). This model opens in normal mode.

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

Create Parameter Object

1. Open `sldrtex_model`.
2. Open the Transfer Fcn block parameters dialog box.
3. Replace the existing value of **Denominator coefficients** (`[1 70 10000]`) with `Dmp`.
4. Click **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: Dmp

Absolute tolerance: auto

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

5 Right-click variable `Dmp` and select **Create Variable: Dmp**.

6 In the **Value** field, select `Simulink.Parameter`.

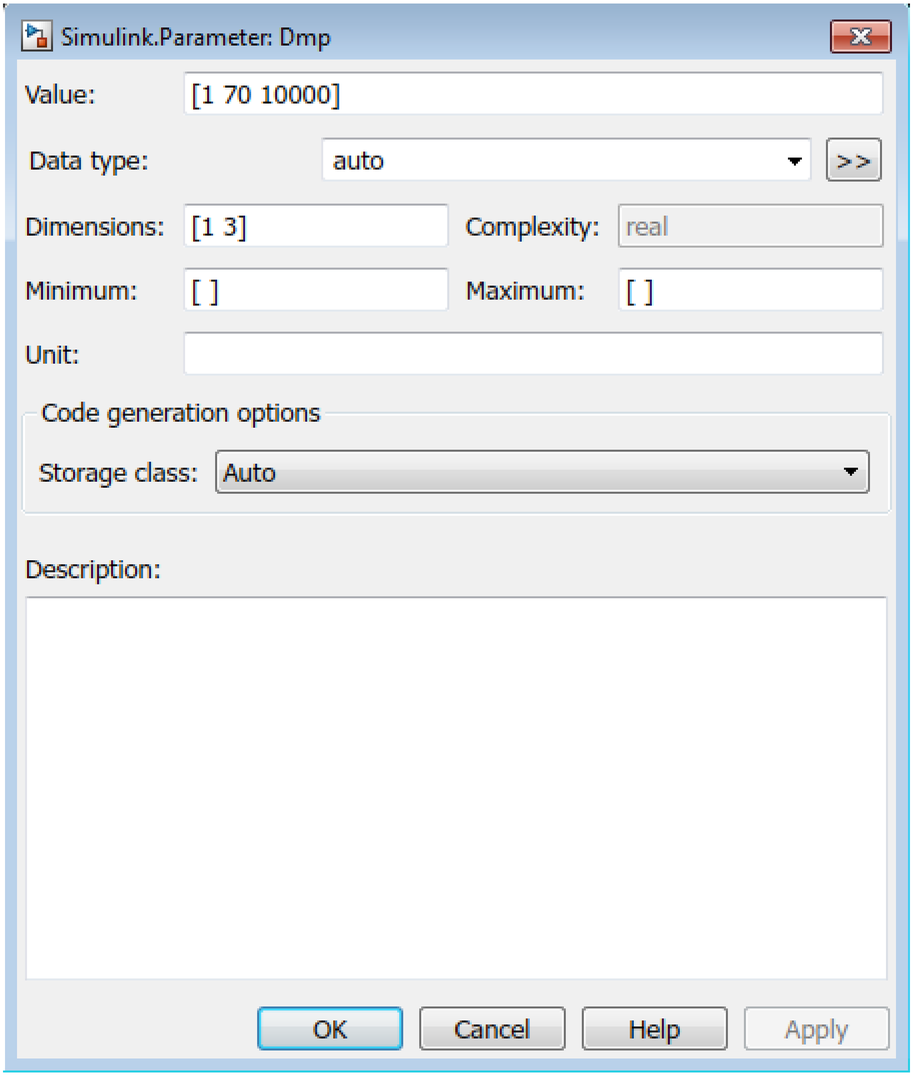
7 In the **Location** field, select **Base Workspace**.

8 Click **Create**.

If the model is already in external 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.



Simulink.Parameter: Dmp

Value: [1 70 10000]

Data type: auto

Dimensions: [1 3]

Complexity: real

Minimum: [ ]

Maximum: [ ]

Unit:

Code generation options

Storage class: Auto

Description:

10 Click **OK**.

11 In the **Block Parameters: Transfer Fcn** dialog box, click **OK**.

Tune Parameter Object

1 Set **Simulation** > **Mode** to `External`.

2 In Simulink Editor, change **Simulation stop time** to `Inf`.

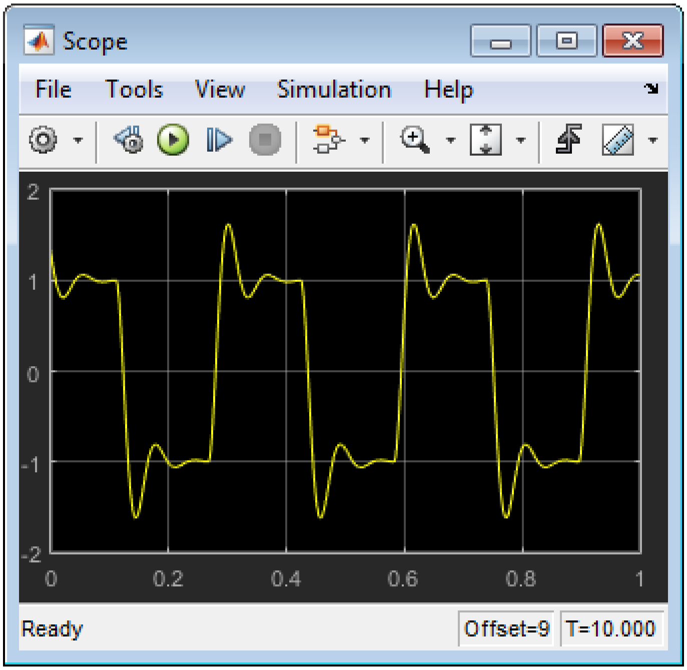
3 Open the Scope block.

4 Open the Transfer Fcn block parameters dialog box.

5 Right-click variable `Dmp` and select **Open variable: Dmp (base workspace)**.

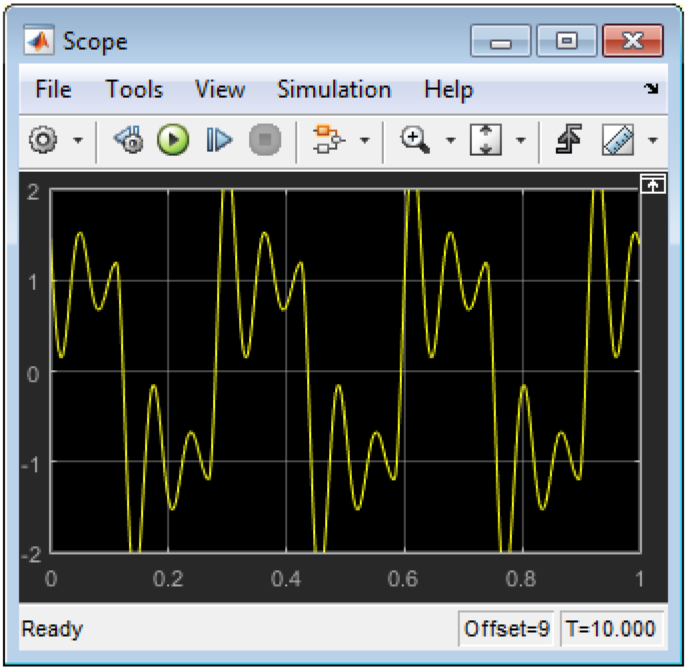
Before you start execution, open this dialog box. You cannot open variable `Dmp` while the real-time application is running.

6 To start execution, on the Simulink Editor toolbar, click the **Run** button ▶.



7 In the **Simulink.Parameter: Dmp** dialog box, change **Value** to `[1 30 10000]` and click **Apply**.

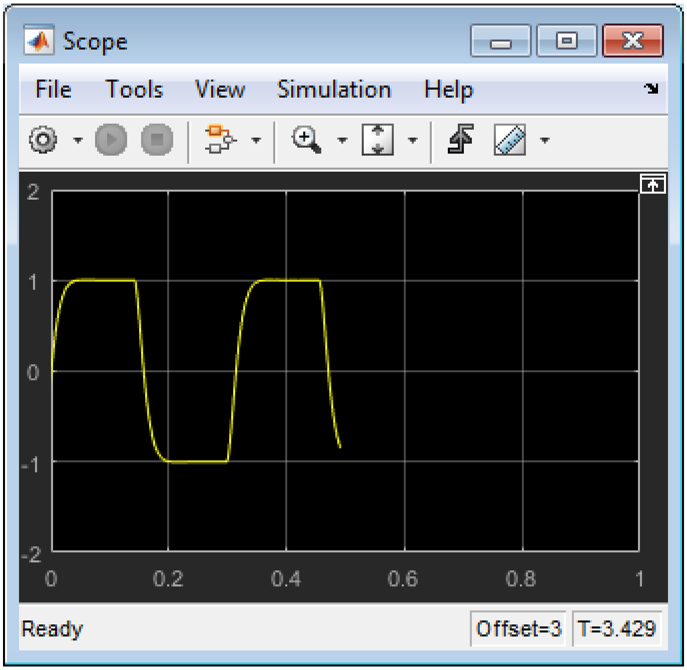
8 Change the active dialog box by clicking in Simulink Editor, and then press **Ctrl-D**.



Ready Offset=9 T=10.000

9 Change **Value** to `[1 180 10000]` and click **Apply**.

10 Change the active dialog box by clicking in Simulink Editor, and then press **Ctrl-D**.



11 Click **Stop**.

More About

- "Share and Reuse Block Parameter Values by Creating Variables"
- "Tunable Block Parameters and MATLAB Variables" on page 3-50

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 normal or accelerator mode, Simulink transfers the new values to the model that is being simulated. In external 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 `sldrtex_inlined` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'sldrt', 'examples', 'sldrtex_inlined'))))`.

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

1. Open `sldrtex_inlined` and the Scope block.

   ```
   model = 'sldrtex_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');
   ```

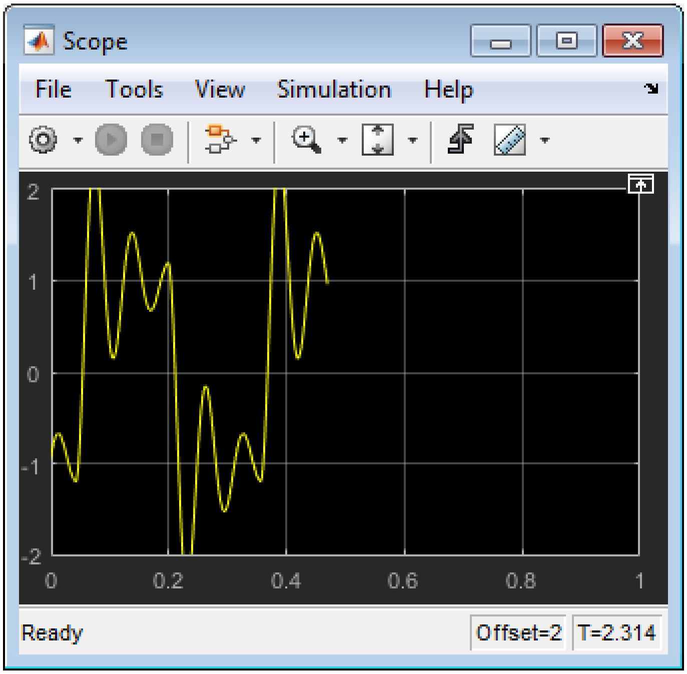
```
set_param(model,'SimulationMode','external')
set_param(model,'SimulationCommand','connect')
set_param(model,'SimulationCommand','start')
```

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

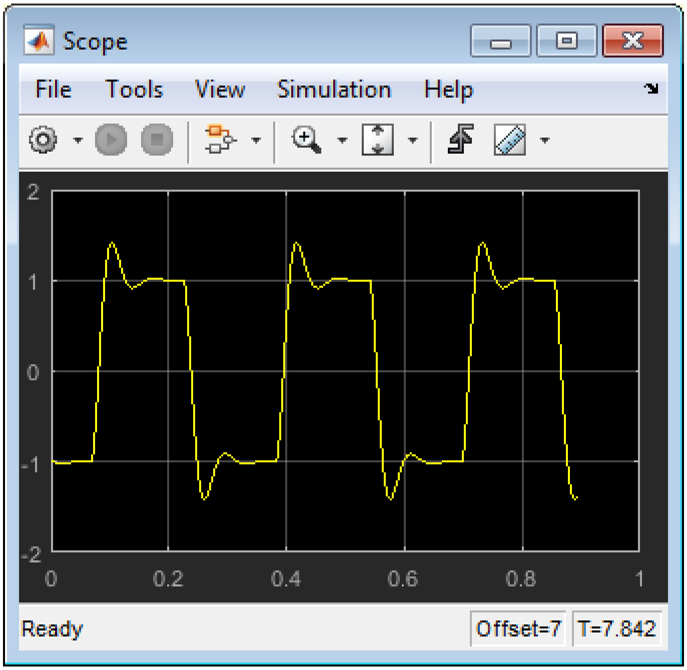
```
for Val = 30 : 30 : 180
    Dmp.Value = [1 Val 10000];
    set_param(model,'SimulationCommand','update')

    pause(2.0)
end
```

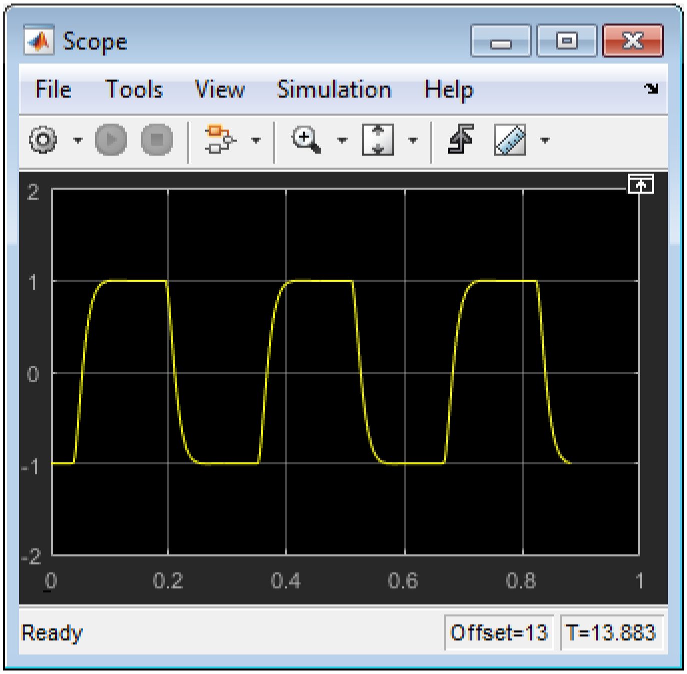
The Scope block shows changes at 30-unit intervals. The figures show key changes.



Val == 30


Scope
File
Tools
View
Simulation
Help
2
1
0
-1
-2
0
0.2
0.4
0.6
0.8
1
Ready
Offset=7
T=7.842

Val == 90



Val == 180

6 Stop execution.

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

More About

- “Store Data in Dictionary Programmatically”
- “Tunable Block Parameters and MATLAB Variables” on page 3-50

4

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

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`.

In this section...

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 board's device driver 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 and 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, but not software readable** — Set jumpers or switches on the physical board, and then 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 or not the board exists in the computer, but you will not be able to 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 in most cases. Details for various types of boards and drivers appear later in this topic.

The following instructions use the HUMUSOFT® AD512 I/O board as an example, configure the board for analog input, and assume that you have physically configured and installed the board in your computer before you add its driver to your model. Customize the steps to achieve the results you need.

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. In the model window, choose **View > Library Browser** to display the Simulink 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.

   The I/O Block Parameters dialog box opens.
5. 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. Select one of the following, as required by the board type:

   - For an ISA bus board, enter a hexadecimal base address. This value must match the base address jumpers or switches set on the physical board. For example, to enter a base address of `0x300`, in the **Address** box type

     ```
     300
     ```

     You can also select the base address by checking boxes **A9** through **A3**.

Note: Support for ISA bus boards will be removed in a future release. Use PCI, PC/104, Compact PCI, or PCMCIA boards instead.

- For 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** on the message box, and again on 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, you must execute the complete installation sequence for each board, just as you would 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. You must specify their logical device numbers manually.

Multiple Driver Blocks for One Board

Once you have used the I/O Block Parameters dialog box to add a board and configure its driver, you can add additional I/O driver blocks that connect to the same board from other locations in the model. To do this, drag the required driver block into the model, open its I/O Block Parameters dialog box, and choose 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 need to specify these parameters only once, when you first add the board and configure its driver. If you subsequently change a parameter in a driver block connected to a board, the same change occurs in the other driver blocks connected to that board.

ISA Bus Board

Most ISA bus I/O boards are preset with a base address of `0x300`. If you are using multiple I/O boards or other boards (for example, network cards) that already use the address `0x300`, you must set your board with another base address.

In the I/O board dialog box, enter the same base address that you set on the physical board. You open the I/O board dialog box from an I/O driver Block Parameters dialog box.

Note: Support for ISA bus boards will be removed in a future release. Use PCI, PC/104, Compact PCI, or PCMCIA boards instead.

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, you should 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, the board might be 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 may be able to 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.

If you want to use an unsupported board that you believe should be supported, contact MathWorks Technical Support at `www.mathworks.com/contact_TS.html`.

PC/104 Board

Most PC/104 bus I/O boards are preset with a base address of `0x300`. If you are using multiple I/O boards or other boards (for example, network cards) that already use the address `0x300`, you must set your board with another base address.

In the I/O board dialog box, enter the same base address that you set on the physical board. You open the I/O board dialog box from an I/O driver Block Parameters dialog box.

Compact PCI Board

Using a compact PCI board (PXI®, PXI Express) requires you to use a compact PC (industrial PC). In addition, you need to 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, the board might be invisible to your computer and to the Simulink Desktop Real-Time software.

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 to 4 and another block for channels 5 to 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.

In this section...

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 model window, choose **View > 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, you need to 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 window, choose **View > Library Browser**.

   The Simulink Library Browser opens.
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. In the Simulink window, click **Display** > **Signals & Ports** > **Wide Nonscalar Lines**.
8. Click **Display** > **Signals & Ports** > **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

For a better understanding of how to specify device settings when using both analog and digital signals, this section uses the Keithley® Metrabyte™ DAS-1601 I/O board as an example. 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 to 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 to 5 volts, 0 to 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 for a single analog input signal that will be 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 — Analog channels are numbered starting with channel 1 and continue until you reach a number corresponding 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]
```

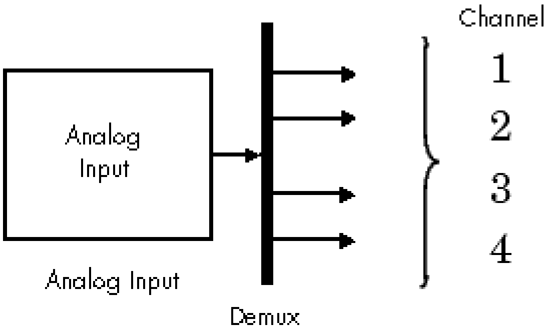
Input vector with single-ended analog — Now, assume your DAS-1601 board is configured to be single-ended analog input. In this case, 16 analog input channels are supported. The complete input vector is

```
[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]
```

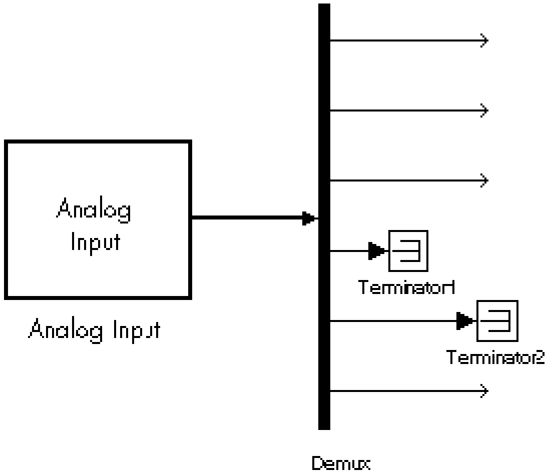
The next figure shows the resulting block diagram.



Do not specify more channels than you actually use in your block diagram. This 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 actually 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 to clearly indicate 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 are actually required by your model.

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.

In this section...

Configure I/O Driver Characteristics

Simulink Desktop Real-Time applications use off-the-shelf 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. Considerable effort is made to minimize the latency between collecting a data point and using the data in the control system algorithm. This is why a board might 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, pressure, and so on. This ability to choose normalized 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 are received by the board, and you choose the block output signal. For example, the voltage range could be set to `0 to +5 V`, and the block output signal could be chosen as `Normalized unipolar`, `Normalized bipolar`, `Volts`, or `Raw`.

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

If you prefer to apply your own scaling factor, you can choose `Normalized unipolar` or `Normalized bipolar`, add a Gain block, and add an offset to convert to a meaningful value in your model.

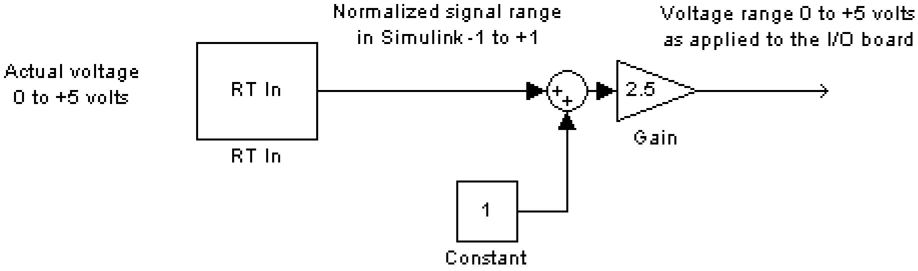
If you prefer unrounded integer values from the analog-to-digital conversion process, you can choose `Raw`.

0 to +5 Volts and Normalized Bipolar

From the Input range list, choose `0 to +5 V`, and from the `Block output signal` list, choose `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 to 1] normalized + 1) * 2.5
```

In your block diagram, you can do this as follows.

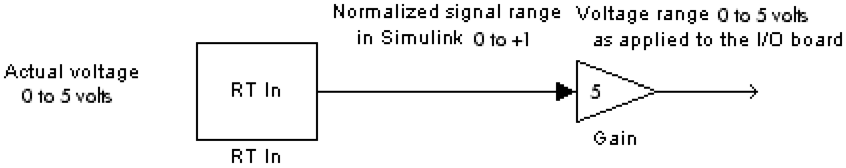


0 to +5 Volts and Normalized Unipolar

From the `Input range` list, choose `0 to +5 V`, and from the `Block output signal` list, choose `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 to 1] normalized * 5.0
```

In your block diagram, you can do this as follows.

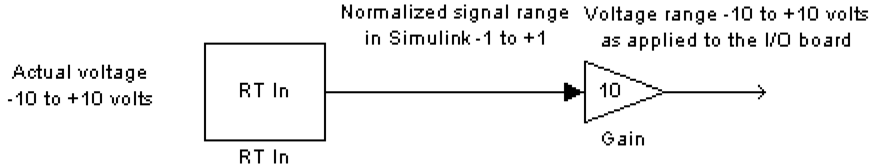


-10 to +10 Volts and Normalized Bipolar

From the `Input range` list, choose `-10 to +10 V`, and from the `Block output signal` list, choose `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 to +1] normalized * 10
```

In your block diagram, you can do this as follows.

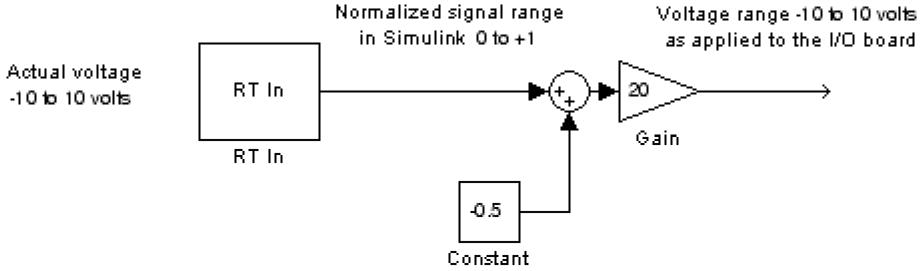


-10 to +10 Volts and Normalized Unipolar

From the `Input range` list, choose `-10 to +10 V`, and from the `Block output signal` list, choose `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 to 1] normalized - 0.5) * 20
```

In your block diagram, do this as follows.



Normalize Scaling for Analog Outputs

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

If the voltage range on the D/A converter is set to `0 to +5 volts`, and the `Block input signal` is chosen as `Normalized bipolar`, then 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.

A voltage range on the D/A converter is set to `-10 to +10 volts`, and the `Block input signal` is chosen as `Normalized bipolar`, then 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.

This may require that you adjust your signal amplitudes using a Gain block, Constant block, and Summer block depending on the selected voltage range.

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, support libraries, and so on.
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 web site:

   `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`.

5

Troubleshooting

Solutions have been worked out for some common errors and problems that can occur when you are using Simulink Desktop Real-Time software.

Failure to Connect to Target

Possible Problem — When trying to run a model, the Simulation Errors dialog box displays:

```
Error occurred while executing External Mode MEX-file

     'sldrtext': Incorrect version
```

Solution — This indicates that the Simulink Desktop Real-Time kernel may be out of date. Type `rtwho` at the MATLAB prompt. If you see a message like this, install the current version:

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

Possible Problem — When trying to connect to the target, the Simulation Errors dialog box displays:

```
Checksum mismatch. Target code needs to be rebuilt.
```

Solution — This indicates that the model structure has changed since the last time code was generated. You must rebuild the real-time application. If your model fails to build, delete the `.mk` and `.obj` files from the Simulink Coder project folder, and then select **Build** from the **Tools** menu.

More About

- "Install Real-Time Kernel" on page 2-4

Scope Output Delayed or Missing

During an external mode run, you may notice either slow updates of Scope blocks or a complete failure to plot data in Scope blocks. This could indicate that the real-time application sample time is near the lower threshold for your 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, so a small sample time may allow the application to run but 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.

To check the sample time, select a larger sample time for your application. You should also change the sample time of any I/O drivers to be the same as the new application sample time, or 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` second, and confirm that your system runs and plots are displayed. Then decrease the sample time until you can both display scopes and meet your accuracy and response time requirements. You must update the I/O driver sample time and rebuild the application after changing the application sample time.

Plots Not Visible in Simulink Scope Block

Prior to an external mode run, you must specify the following for data to plot in a Simulink Scope block:

1. Select **Simulation** > **Mode** > **External**.
2. Select one or more signals for capture (designated with `X`) in the External Signal & Triggering dialog box from the External Mode Control Panel.
3. Set **Duration * Fixed Step Size** close to or less than the `X` range in the Scope block
4. Select required mode (`one-shot` or `normal`)
5. Configure signal levels to allow triggering
6. Set `Y` range on Simulink Scope block axes to large enough to span the signal amplitude
7. Set `X` range to large enough to provide required time resolution
8. Set **Arm when connect to target** in the External Signal & Triggering dialog box or **Arm Trigger** in the External Mode Control Panel.
9. Click the Run button ▶ on the toolbar.

If you cannot see signals plotted in your Simulink Scope blocks after carrying out these steps, your system might have too little CPU time available between sample intervals to transfer data back to MATLAB, where the plotting is performed. To test for this condition, run one of the example models or run your model at a significantly slower sample rate.

S-Functions Using Math Functions

Possible problem — When you create your own S-functions that include math functions, the S-functions compile, but you cannot build the application.

Solution — Add the Simulink Desktop Real-Time header to your S-function. For example, add

```
#include <math.h>
#include "sldrtkernel.h"
```

The header `#include <math.h>` must precede the header `#include "sldrtkernel.h"`.

Building Older Models

If you are building an older model with Simulink Desktop Real-Time software, you might get a message like the following:

```
"Simulink Coder utilizes device specific information (e.g.,
microprocessor word sizes) to reproduce a bit true representation
of the diagram. This information is not specified in this model.
If you continue, Simulink Coder will use a 32-bit generic
target setting."
```

This is simply a warning, and you can ignore the message. To eliminate this message, you can use the `sldrtconfigset` command, as follows:

```
sldrtconfigset('<model_name>')
```

Vendor Software Required

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 this 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.

If you encounter such instances, 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.

Unsupported Models

In external mode, Simulink Desktop Real-Time does not support Simscape and Simscape Driveline models.

Custom I/O Driver Blocks

Custom I/O Driver Basics

In this section...

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, you should use only C functions supported by Simulink Desktop Real-Time software. Functions that use the operating system are not supported by Simulink Desktop Real-Time. This 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, vfprintf, vprintf, vsprintf, fscanf, scanf, sscanf
- **Process management** — spawn, exit, abort, atexit
- **Signals and exceptions** — signal, longimp, 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 standard applications or any 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
```