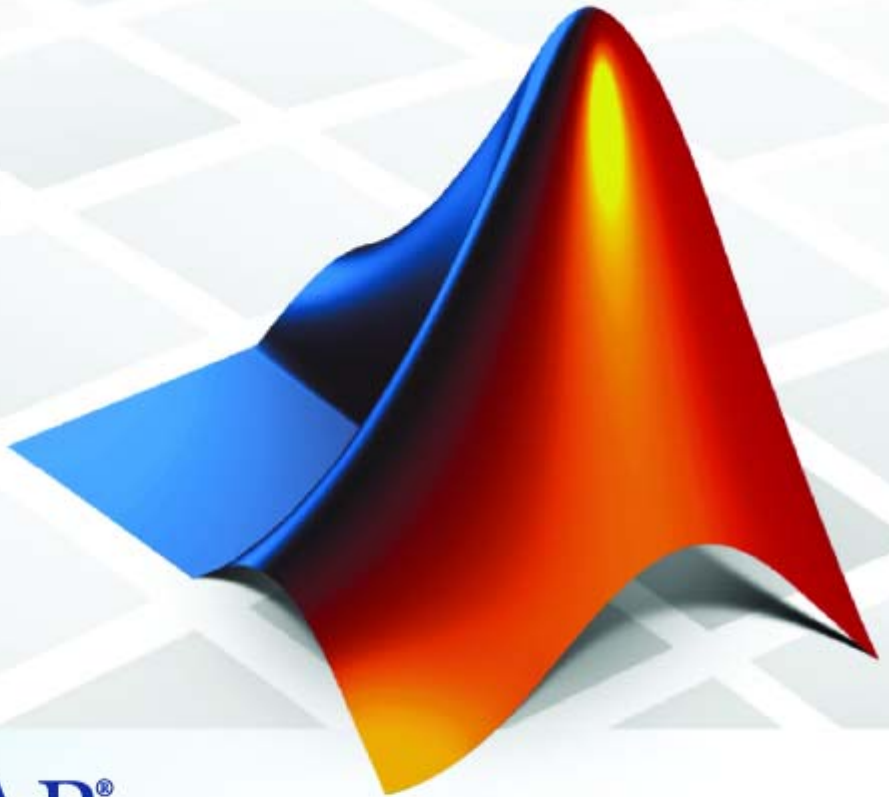


Real-Time Windows Target 2

User's Guide



MATLAB[®]
& **SIMULINK[®]**

How to Contact The MathWorks



www.mathworks.com
comp.soft-sys.matlab
www.mathworks.com/contact_TS.html

Web
Newsgroup
Technical Support



suggest@mathworks.com
bugs@mathworks.com
doc@mathworks.com
service@mathworks.com
info@mathworks.com

Product enhancement suggestions
Bug reports
Documentation error reports
Order status, license renewals, passcodes
Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



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

For contact information about worldwide offices, see the MathWorks Web site.

Real-Time Windows Target User's Guide

© COPYRIGHT 1999–2007 by The MathWorks, Inc.

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

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

Trademarks

MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks, and SimBiology, SimEvents, and SimHydraulics are trademarks of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

January 1999	First printing	New for Version 1.0 (Release 11.0)
January 2000	Second printing	Revised for Version 1.5 (Release 11.1+)
September 2000	Third printing	Revised for Version 2.0 (Release R12)
June 2001	Online only	Revised for Version 2.1 (Release R12.1)
July 2002	Online only	Revised for Version 2.2 (Release 13)
June 2004	Fourth printing	Revised for Version 2.5 (Release 14)
October 2004	Fifth printing	Revised for Version 2.5.1 (Release 14SP1)
March 2005	Online only	Revised for Version 2.5.2 (Release 14SP2)
September 2005	Online only	Revised for Version 2.6 (Release 14SP3)
March 2006	Online only	Revised for Version 2.6.1 (Release 2006a)
September 2006	Online only	Revised for Version 2.6.2 (Release 2006b)
March 2007	Online only	Revised for Version 2.7 (Release 2007a)

Getting Started

1

What Is Real-Time Windows Target?	1-2
Expected User	1-3
Features	1-4
Real-Time Kernel	1-4
Real-Time Application	1-6
Signal Acquisition and Analysis	1-6
Parameter Tuning	1-7
Hardware Environment	1-9
PC-Compatible Computer	1-9
Input/Output Driver Support	1-9
Software Environment	1-11
Non-Real-Time Simulation	1-11
Real-Time Execution	1-11
Development Process	1-12
System Concepts	1-13
Simulink External Mode	1-13
Data Buffers and Transferring Data	1-14

Installation and Configuration

2

Required Products	2-2
MATLAB	2-2
Simulink	2-2
Real-Time Workshop	2-3

Related Products	2-4
System Requirements	2-5
Hardware Requirements	2-5
Software Requirements	2-6
Real-Time Windows Target Installed Files	2-7
Initial Working Directory	2-9
Setting the Working Directory from the Desktop Icon	2-9
Setting the Working Directory from MATLAB	2-9
Real-Time Windows Target Kernel	2-10
Installing the Kernel	2-10
Uninstalling the Kernel	2-11
Testing the Installation	2-14
Running the Model rtvdp.mdl	2-14
Displaying Status Information	2-16
Detecting Excessive Sample Rates	2-17
Demo Library	2-18

Basic Procedures

3

Simulink Model	3-2
Creating a Simulink Model	3-2
Entering Configuration Parameters for Simulink	3-6
Entering Scope Parameters for Signal Tracing	3-7
Running a Non-Real-Time Simulation	3-10
Specifying a Default Real-Time Windows Target Configuration Set	3-11
Real-Time Application	3-13
Entering Simulation Parameters for Real-Time Workshop	3-13
Entering Scope Parameters for Signal Tracing	3-16
Creating a Real-Time Application	3-18

Entering Additional Scope Parameters for Signal Tracing	3-19
Running a Real-Time Application	3-21
Running a Real-Time Application from the MATLAB Command Line	3-24
Signal Logging to the MATLAB Workspace	3-26
Entering Scope Parameters	3-26
Entering Signal and Triggering Properties	3-28
Plotting Logged Signal Data	3-31
Signal Logging to a Disk Drive	3-33
Entering Scope Parameters	3-33
Entering Signal and Triggering Properties	3-36
Entering Data Archiving Parameters	3-38
Plotting Logged Signal Data	3-40
Parameter Tuning	3-43
Types of Parameters	3-43
Changing Model Parameters	3-44

Advanced Procedures

4

I/O Boards	4-2
Installing and Configuring I/O Boards and Drivers	4-2
ISA Bus Board	4-6
PCI Bus Board	4-7
PC/104 Board	4-8
Compact PCI Board	4-8
PCMCIA Board	4-8
I/O Driver Blocks	4-9
Real-Time Windows Target Library	4-9
Simulink Library	4-11
Analog Input Block	4-12
Analog Output Block	4-14
Digital Input Block	4-16
Digital Output Block	4-17

Counter Input Block	4-20
Encoder Input Block	4-22
Other Input and Other Output Blocks	4-24
Output Signals from an I/O Block	4-24
Variations with Channel Selection	4-25
Using Analog I/O Drivers	4-29
I/O Driver Characteristics	4-29
Normalized Scaling for Analog Inputs	4-30

Troubleshooting

5

Building Older Models	5-2
Plots Not Visible in Simulink Scope Block	5-3
Failure to Connect to Target	5-4
Sample Time Too Fast	5-5
S-Functions Using Math Functions	5-6
Restricted Space for S-Function Local Variables	5-7

Custom I/O Driver Blocks Reference

A

I/O Register Access from S-Functions Limitation	A-2
Incompatibility with Win32 API Calls	A-3
Unsupported C Functions	A-4

Supported C Functions	A-5
------------------------------------	------------

Examples

B

Simulink Model Examples	B-2
Real-Time Application Examples	B-2
Signal Logging to MATLAB Workspace Examples	B-2
Signal Logging to Disk Drive Examples	B-2
Parameter Tuning Examples	B-3
I/O Board Examples	B-3

Index

Getting Started

Real-Time Windows Target has many features. An introduction to these features and the Real-Time Windows Target software environment will help you develop a model for working with Real-Time Windows Target.

What Is Real-Time Windows Target? (p. 1-2)	A PC solution for prototyping and testing real-time systems
Features (p. 1-4)	Real-time kernel, real-time application, signal acquisition and analysis, and parameter tuning
Hardware Environment (p. 1-9)	PC-compatible computer and I/O support boards
Software Environment (p. 1-11)	Non-real-time simulation of Simulink models and real-time execution of applications
System Concepts (p. 1-13)	Simulink external mode and data buffers

What Is Real-Time Windows Target?

Real-Time Windows Target is a PC solution for prototyping and testing real-time systems. It is an environment where you use a single computer as a host and target.

In this environment you use your desktop or laptop PC with MATLAB®, Simulink®, and Stateflow® (optional) to create models using Simulink blocks and Stateflow diagrams.

After creating a model and simulating it with Simulink in normal mode, you can generate executable code with Real-Time Workshop®, Stateflow Coder (optional), and the Open Watcom C/C++ compiler. Then you can run your application in real time with Simulink external mode.

Integration between Simulink external mode and Real-Time Windows Target allows you to use your Simulink model as a graphical user interface for

- **Signal visualization** — Use the same Simulink Scope blocks that you use to visualize signals during a non-real-time simulation to visualize signals while running a real-time application.
- **Parameter tuning** — Use the Block Parameter dialog boxes to change parameters in your application while it is running in real time.

Note Opening a dialog box for a source block causes Simulink to pause. While Simulink is paused, you can edit the parameter values. You must close the dialog box to have the changes take effect and allow Simulink to continue.

Typical applications for Real-Time Windows Target include

- **Real-time control** — Create a prototype of automotive, computer peripheral, and instrumentation control systems.
- **Real-time hardware-in-the-loop simulation** — Create a prototype of controllers connected to a physical plant. For example, the physical plant could be an automotive engine. Create a prototype of a plant connected to

an actual controller. For example, the prototyped plant could be an aircraft engine.

- **Education** — Teach concepts and procedures for modeling, simulating, testing real-time systems, and iterating designs.

Expected User

To benefit from reading this book, you should be familiar with

- Using Simulink and Stateflow to create models as block diagrams, and simulating those models in Simulink
- The concepts and use of Real-Time Workshop to generate executable code

When using Real-Time Workshop and Real-Time Windows Target, you do not need to program in C or other low-level programming languages to create and test real-time systems.

If You Are a New User — Begin with Chapter 1, “Getting Started”. This chapter gives you an overview of Real-Time Windows Target features and the development environment. Next, read and try the examples in Chapter 3, “Basic Procedures”.

If You Are an Experienced Real-Time Windows Target User — We suggest you review the sections on signal tracing and signal logging in Chapter 3, “Basic Procedures”. After you are familiar with using Real-Time Windows Target, read how to add I/O drivers to your Simulink model in Chapter 4, “Advanced Procedures”.

Features

- “Real-Time Kernel” on page 1-4
- “Real-Time Application” on page 1-6
- “Signal Acquisition and Analysis” on page 1-6
- “Parameter Tuning” on page 1-7

The Real-Time Windows Target software environment includes many features to help you prototype and test real-time applications.

Real-Time Kernel

Real-Time Windows Target uses a small real-time kernel to ensure that the real-time application runs in real time. The real-time kernel runs at CPU ring zero (privileged or kernel mode) and uses the built-in PC clock as its primary source of time:

- **Timer interrupt** — The kernel intercepts the interrupt from the PC clock before the Windows operating system receives it. This blocks any calls to the Windows operating system. Because of this, **you cannot use Win32 calls in your C-code S-function.**

The kernel then uses the interrupt to trigger the execution of the compiled model. As a result, the kernel is able to give the real-time application the highest priority available.

To achieve precise sampling, the kernel reprograms the PC clock to a higher frequency. Because the PC clock is also the primary source of time for the Windows operating system, the kernel sends a timer interrupt to the operating system at the original interrupt rate.

Technically, the kernel is provided as a kernel-mode driver on Windows 2000 and Windows XP.

- **Scheduler** — The timer interrupt clocks a simple scheduler that runs the executable. The number of tasks is equal to the number of sampling periods in the model with multitasking mode. With single-tasking mode, there is only one task. The maximum number of tasks is 32, and faster tasks have higher priorities than slower tasks. For example, a faster task can interrupt a slower task.

During execution, the executable stores data in buffers. Later, the data in these buffers is retrieved by the Scope block. The scheduling, data storing, data transferring, and running the executable all run at CPU ring zero.

- **Communication with hardware** — The kernel interfaces and communicates with I/O hardware using I/O driver blocks, and it checks for proper installation of the I/O board. If the board has been properly installed, the drivers allow your real-time application to run.

The Analog Input, Analog Output, Digital Input, Digital Output, Counter Input, and Encoder Input blocks call the drivers for input and output. You can choose to have a driver block use values equal to voltage, normalize values from 0 to +1, normalize values from -1 to +1, or use the raw integer values from the A/D or D/A conversion press. Drivers also run at CPU ring zero.

- **Simulink external mode** — Communication between Simulink and the real-time application is through the Simulink external mode interface module. This module talks directly to the real-time kernel, and is used to start the real-time application, change parameters, and retrieve scope data.

Opening a dialog box for a source block causes Simulink to pause. While Simulink is paused, you can edit the parameter values. You must close the dialog box to have the changes take effect and allow Simulink to continue.

- **Built-in C compiler** — Real-Time Windows Target applications are compiled with the Open Watcom C/C++ compiler. For your convenience, this compiler is shipped with Real-Time Windows Target. No other third-party compilers are necessary or can be used.

Note Real-Time Windows Target always uses the Open Watcom C/C++ compiler, even if you have specified some other compiler using the `mex -setup` command. Real-Time Windows Target cannot be configured to use a compiler other than Open Watcom C/C++.

Real-Time Application

The real-time application runs in real time on your PC computer and has the following characteristics:

- **Compiled code** — Created from the generated C-code using the Open Watcom C/C++ compiler.
- **Relation to your Simulink model** — The executable contains a binary form of all Simulink model components, connections between blocks, time dependencies, and variables in the Simulink blocks.
- **Relation to the kernel** — The executable must be loaded and executed directly by the Real-Time Windows Target kernel. It cannot be executed without the kernel.

The kernel runs as a kernel-mode driver, intercepts timer interrupts from the PC clock, maintains clock signals for the Windows operating system, and ensures real-time execution of the real-time application. As a result, both the kernel and the real-time application run at CPU ring zero.

- **Checksum** — The Simulink model and the executable contain a checksum value. The kernel uses this checksum value to determine if the Simulink model structure, at the time of code generation, is consistent with the real-time application structure during execution. This ensures that when you change parameters during an execution, the mapping of Simulink model parameters to the memory locations in the real-time application is correct.

If you make structural changes to your Simulink model, the Simulink checksum value will not match the executable checksum value. You will have to rebuild your executable before you can connect it to your Simulink model.

Signal Acquisition and Analysis

You can acquire, display, and save signals by using Simulink Scope blocks and Simulink external mode. This lets you observe the behavior of your model during a simulation or your application while it runs in real time.

You can acquire signal data while running your real-time applications using

- **Signal Tracing** — Process of acquiring and visualizing signals during a real-time run. It allows you to acquire signal data and visualize it on your computer while the executable is running.
- **Signal Logging** — Process for acquiring signal data during a real-time run. After the run reaches its final time or you manually stop the run, you can plot and analyze the data.

You can save (log) data to variables in the MATLAB workspace or save data to your disk drive with MAT-files.

Signal logging differs from signal tracing. With signal logging you can only look at a signal after a run is finished.

For more information, see “Signal Logging to the MATLAB Workspace” on page 3-26 and “Signal Logging to a Disk Drive” on page 3-33.

Parameter Tuning

Change the parameters in your Simulink model and observe the effect of those changes during a simulation or while running an application in real time.

Simulink external mode — You use Simulink external mode to connect your Simulink block diagram to your real-time application. The block diagram becomes a graphical user interface (GUI) to that executable.

Simulink external mode allows you to change parameters by editing the block diagram while running Simulink in external mode. New parameter values are automatically transferred to the real-time application while it is running.

Note Opening a dialog box for a source block causes Simulink to pause. While Simulink is paused, you can edit the parameter values. You must close the dialog box to have the changes take effect and allow Simulink to continue.

There are different types of model parameters that you can change while running your real-time application. For example, parameters include the amplitude of a gain and the frequency of a sine wave. After you connect your

real-time application to your Simulink model, you can change parameters. You can change these parameters before or while your real-time application is running by using one of the following methods:

- **Block parameters** — Change values in the dialog boxes associated with the Simulink blocks.
- **Block parameters for masked subsystems** — Change values in user-created dialog boxes associated with a subsystem.
- **MATLAB variables** — Create MATLAB variables that represent Simulink block parameters, and then change parameter values by entering the changes through the MATLAB command line.

For more information about parameter tuning, see “Parameter Tuning” on page 3-43.

Hardware Environment

- “PC-Compatible Computer” on page 1-9
- “Input/Output Driver Support” on page 1-9

The hardware environment consists of a PC-compatible computer and I/O boards.

PC-Compatible Computer

You can use any PC-compatible computer that runs Windows 2000 or Windows XP.

Your computer can be a desktop, laptop, or notebook PC.

Input/Output Driver Support

Real-Time Windows Target uses standard and inexpensive I/O boards for PC-compatible computers. When running your models in real time, Real-Time Windows Target captures the sampled data from one or more input channels, uses the data as inputs to your block diagram model, immediately processes the data, and sends it back to the outside world through an output channel on your I/O board.

I/O Boards

I/O boards — Real-Time Windows Target supports a wide range of I/O boards. The list of supported I/O boards includes ISA, PCI, and PCMCIA boards. This includes analog-to-digital (A/D), digital-to-analog (D/A), digital inputs, digital outputs, and encoder inputs. In total, over 200 I/O boards are currently supported.

Note Some of the functions on a board might not be supported by Real-Time Windows Target. Check the MathWorks Web site for an updated list of supported boards and functions at Supported I/O Boards.

I/O Driver Block Library

Real-Time Windows Target provides a custom Simulink block library. The I/O driver block library contains universal drivers for supported I/O boards. These universal blocks are configured to operate with the library of supported drivers. This allows easy location of driver blocks and easy configuration of I/O boards.

You drag and drop a universal I/O driver block from the I/O library the same way as you would from a standard Simulink block library. And you connect an I/O driver block to your model just as you would connect any standard Simulink block.

You create a real-time application in the same way as you create any other Simulink model, by using standard blocks and C-code S-functions. You can add input and output devices to your Simulink model by using the I/O driver blocks from the `rtwinlib` library provided with Real-Time Windows Target. This library contains the following blocks:

- Analog Input
- Analog Output
- Digital Input
- Digital Output
- Counter Input
- Encoder Input

Real-Time Windows Target provides driver blocks for more than 200 I/O boards. These driver blocks connect the physical world to your real-time application:

- Sensors and actuators are connected to I/O boards.
- I/O boards convert voltages to numerical values and numerical values to voltages.
- Numerical values are read from or written to I/O boards by the I/O drivers.

Software Environment

- “Non-Real-Time Simulation” on page 1-11
- “Real-Time Execution” on page 1-11
- “Development Process” on page 1-12

The software environment is a place to design, build, and test an application in nonreal time and real time.

Non-Real-Time Simulation

You create a Simulink model and use Simulink in normal mode for non-real-time simulation on your PC computer.

Simulink model — Create block diagrams in Simulink using simple drag-and-drop operations, and then enter values for the block parameters and select a sample rate.

Non-real-time simulation — Simulink uses a computed time vector to step your Simulink model. After the outputs are computed for a given time value, Simulink immediately repeats the computations for the next time value. This process is repeated until it reaches the stop time.

Because this computed time vector is not connected to a hardware clock, the outputs are calculated in nonreal time as fast as your computer can run. The time to run a simulation can differ significantly from real time.

Real-Time Execution

For real-time execution on your PC computer, create a real-time application and use Simulink in external mode. Real-Time Workshop, Real-Time Windows Target, and the Open Watcom C/C++ compiler produce an executable that the kernel can run in real time. This real-time application uses the initial parameters available from your Simulink model at the time of code generation.

If you use continuous-time components in your model and generate code with Real-Time Workshop, you must use a fixed-step integration algorithm. Real-Time Windows Target provides the necessary software that uses the

real-time resources on your computer hardware. Based on your selected sample rate, Real-Time Windows Target uses interrupts to step your application in real time at the proper rate. With each new interrupt, the executable computes all of the block outputs from your model.

Development Process

In the Real-Time Windows Target environment, you use your desktop PC with MATLAB, Simulink, Real-Time Workshop, and Real-Time Windows Target to

- 1 Design a control system** — Use MATLAB and Control System Toolbox to design and select the system coefficients for your controller.
- 2 Create a Simulink model** — Use Simulink blocks to graphically model your physical system.
- 3 Run a simulation in nonreal time** — Check the behavior of your model before you create a real-time application. For example, you can check the stability of your model.
- 4 Create a real-time application** — Real-Time Workshop generates C code from your Simulink model. The Open Watcom C/C++ compiler compiles the C code to an executable that runs with the Real-Time Windows Target kernel.
- 5 Run an application in real time** — Your desktop PC is the target computer to run the real-time application.
- 6 Analyze and visualize signal data** — Use MATLAB functions to plot data saved to the MATLAB workspace or a disk.

Note Although Real-Time Windows Target applications run on the same hardware as Windows, the Real-Time Windows Target kernel and the Win32 kernel are incompatible. When a Real-Time Windows Target application includes externally created code, such as a custom I/O driver block or a user-supplied S-function, the code cannot access any Win32 function. For more information, see “Incompatibility with Win32 API Calls” on page A-3.

System Concepts

- “Simulink External Mode” on page 1-13
- “Data Buffers and Transferring Data” on page 1-14

A more detailed understanding of Real-Time Workshop and Real-Time Windows Target can help you when creating and running your real-time applications.

Simulink External Mode

External mode requires a communications interface to pass parameters external to Simulink, and on the receiving end, the same communications protocol must be used to accept new parameter values and insert them in the proper memory locations for use by the real-time application. In some Real-Time Workshop targets such as Tornado/VME targets, the communications interface uses TCP/IP protocol. In the case of Real-Time Windows Target, the host computer also serves as the target computer. Therefore, only a virtual device driver is needed to exchange parameters between MATLAB and Simulink memory space and memory that is accessible by the real-time application.

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.

Note that if you open a source block to change parameters, the simulation will pause while the block dialog box is open. You must close the dialog by clicking **OK**, which will resume the simulation.

As a user of Real-Time Windows Target, you will find that the requirements for setup are minimal. You start by enabling external mode. You then choose the RTW system target file from the Configuration Parameters dialog

Real-Time Workshop tab. The MEX-file interface is automatically selected when you choose the target file. Then, after you have built the real-time application, you are ready for external mode operation.

Data Buffers and Transferring Data

At each sample interval of the real-time application, Simulink stores contiguous data points in memory until a data buffer is filled. Once the data buffer is filled, Simulink suspends data capture while the data is transferred back to MATLAB through Simulink external mode. Your real-time application, however, continues to run. 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 to Simulink, 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-filenames 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

Real-Time Windows Target requires the installation of MATLAB, Simulink, Real-Time Workshop, and the Real-Time Windows Target kernel. Also, make sure you set your working directory outside the MATLAB root directory.

Required Products (p. 2-2)

MATLAB, Simulink, Real-Time Workshop, and Real-Time Windows Target

Related Products (p. 2-4)

The MathWorks provides several products that are especially relevant to the kinds of tasks you can perform with Real-Time Windows Target.

System Requirements (p. 2-5)

Use any PC-compatible computer with MATLAB, Simulink, Real-Time Workshop, and Real-Time Windows Target

Real-Time Windows Target Installed Files (p. 2-7)

Describes installed files that are unique to Real-Time Windows Target

Initial Working Directory (p. 2-9)

Select a directory outside the MATLAB root directory

Real-Time Windows Target Kernel (p. 2-10)

Install the kernel after installing Real-Time Windows Target

Testing the Installation (p. 2-14)

Use the Simulink model `rtvdp.mdl` to test the build process and a real-time application

Required Products

- “MATLAB” on page 2-2
- “Simulink” on page 2-2
- “Real-Time Workshop” on page 2-3

Real-Time Windows Target is a self-targeting system where the host and the target computer are the same computer. You can install it on a PC-compatible computer running Windows 2000 or Windows XP. Real-Time Windows Target requires the products described in this section.

MATLAB

MATLAB provides the design and analysis tools that you use when creating Simulink block diagrams.

MATLAB documentation — For information on using MATLAB, see *Getting Started with MATLAB*, which explains how to work with data and how to use the functions supplied with MATLAB. For a reference describing the functions supplied with MATLAB, see *MATLAB Function Reference*.

Simulink

Simulink 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. C code S-functions are supported by Real-Time Workshop.

Unsupported Simulink blocks — You can use Real-Time Windows Target with most Simulink blocks including discrete-time and continuous-time systems. Real-Time Windows Target does not support blocks that do not run in real time nor does it support To File blocks.

Limitations with Real-Time Workshop — When you use a continuous-time system and generate code with Real-Time Workshop, you must use a fixed-step integration algorithm. However, M-code S-functions are not supported.

Real-Time Windows Target I/O driver blocks — With Real-Time Windows Target, you can remove the physical system model and replace it

with I/O driver blocks connected to your sensors and actuators. The Real-Time Windows Target I/O library supports more than 200 boards.

Note Some of the functions on a board may not be supported by Real-Time Windows Target. Check the MathWorks Web site for an updated list of supported boards and functions at Supported I/O Boards.

Simulink documentation — For information on Simulink, see *Using Simulink*, which explains how to connect blocks to build models and change block parameters. It also provides a reference that describes each block in the standard Simulink library.

Real-Time Workshop

Real-Time Workshop provides the utilities to convert your Simulink models into C code, and then, with the Open Watcom C/C++ compiler, compile the code into a real-time executable.

Real-Time Windows Target 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, Real-Time Workshop has other code formats that generate the more compact code needed for embedded applications.

Real-Time Workshop documentation — For information on code generation, see the *Real-Time Workshop User's Guide*.

Related Products

The MathWorks provides several products that are especially relevant to the kinds of tasks you can perform with Real-Time Windows Target.

For more information about any of these products, see either

- The online documentation for that product if it is installed on your system
- The MathWorks Web site, at <http://www.mathworks.com/products/rtwt/related.jsp>.

System Requirements

- “Hardware Requirements” on page 2-5
- “Software Requirements” on page 2-6

Real-Time Windows Target requires a PC-compatible computer.

Hardware Requirements

The following table lists the minimum hardware resources Real-Time Windows Target requires on your computer.

Hardware	Description
CPU	Pentium or higher in a desktop, laptop, or compact PCI or PC104 industrial computer
Peripherals	Hard disk drive with 16 megabytes of free space Data acquisition board (for a list of supported boards, see Supported I/O Boards) DVD drive
RAM	128 megabytes minimum, 256 megabytes recommended

When you are using a laptop computer, Real-Time Windows Target is a portable environment where your computer uses PCMCIA cards to interface to real world devices.

Software Requirements

Real-Time Windows Target 2.7 has certain product prerequisites that must be met for proper installation and execution.

The following table lists the software you need to install on your computer to run Real-Time Windows Target.

- Supported Windows OS
- MATLAB 7.4
- Simulink 6.6
- Real-Time Workshop 6.6
- Real-Time Windows Target 2.7

Real-Time Windows Target Installed Files

You can install Real-Time Windows Target as part of the regular installation process documented in MathWorks installation guides. This section describes installed files that are unique to Real-Time Windows Target. When using the product, you may find it helpful to know where these files are located.

- **MATLAB working directory** — Simulink models (`model.mdl`) and the Real-Time Windows Target executable (`model.rwd`)

Note Select a working directory outside the MATLAB root. See “Initial Working Directory” on page 2-9.

- **Real-Time Workshop project directory** — The Real-Time Workshop C-code files (`model.c`, `model.h`) are in a subdirectory called `model_rtwin`.
- **Real-Time Windows Target Files** — The files included with Real-Time Windows Target are located in the directory

```
matlabroot\toolbox\rtw\targets\rtwin
```

- **Open Watcom C/C++ compiler directory** — The Open Watcom C/C++ compiler files are located in a subdirectory called `openwat`.

Real-Time Windows Target provides files to help Real-Time Workshop generate C code from your Simulink model and compile that code to a real-time executable:

- **System Target File** (`rtwin.tlc`) — Defines the process of generating C code for Real-Time Windows Target.
- **Template Makefile and Makefile** (`rtwin.tmf`, `model_name.mk`) — The template makefile serves as a template for generating the real makefile, which the make utility uses during model compilation. During the automatic build procedure, the make command extracts information from the template makefile `rtwintmf.m` and generates the makefile `model_name.mk`.
- **Make Command** (`make_rtw.m`) — The standard make command supplied with Real-Time Workshop.

Other files provided with Real-Time Windows Target include

- **I/O drivers** (*.rwd) — Binaries for I/O device drivers. Real-Time Windows Target does not link the driver object files with your real-time executable. The drivers are loaded into memory and run by the kernel separately.
- **Simulink external mode interface** (rtwinext.mex*) — MEX-file for communicating between Simulink external mode and the Real-Time Windows Target kernel.

Simulink external mode uses the MEX-file interface module to download new parameter values to the real-time model and to retrieve signals from the real-time model. You can display these signals in Simulink Scope blocks.

- **Kernel install and uninstall commands** (rtwintgt.m, rtwho.m) — M-file scripts to install and uninstall the Real-Time Windows Target kernel and check installation.

Initial Working Directory

- “Setting the Working Directory from the Desktop Icon” on page 2-9
- “Setting the Working Directory from MATLAB” on page 2-9

Set your MATLAB working directory outside the MATLAB root directory. The default MATLAB root directory is `c:\matlabN`, where *N* is the MATLAB version number.

Setting the Working Directory from the Desktop Icon

Your initial working directory is specified in the shortcut file you use to start MATLAB. To change this initial directory, use the following procedure:

- 1 Right-click the MATLAB desktop icon, or from the program menu, right-click the MATLAB shortcut.
- 2 Click **Properties**. In the **Start in** text box, enter the directory path you want MATLAB to use initially outside the MATLAB root directory.
- 3 Click **OK**, and then start MATLAB. To check your working directory, type

```
pwd or cd
```

Setting the Working Directory from MATLAB

Use the following procedure as an alternative, but temporary, procedure for setting your MATLAB working directory:

- 1 In the MATLAB Command Window, type

```
cd c:\mwd
```

- 2 Check the current working directory, type

```
cd
```

MATLAB displays

```
ans = c:\mwd or c:\mwd
```

Real-Time Windows Target Kernel

- “Installing the Kernel” on page 2-10
- “Uninstalling the Kernel” on page 2-11

A key component of Real-Time Windows Target is a real-time kernel that interfaces with the Windows operating system in a way that allows your real-time executable to run at your selected sample rate. The kernel assigns the highest priority of execution to your real-time executable.

Installing the Kernel

During installation, all software for Real-Time Windows Target is copied onto your hard drive. The kernel, although copied to the hard drive, is not automatically installed. Installing the kernel sets up the kernel to start running in the background each time you start your computer.

After you install Real-Time Windows Target, you can install the kernel. You need to install the kernel before you can run a Real-Time Windows Target executable:

- 1** In the MATLAB window, type

```
rtwintgt -install
```

MATLAB displays the message

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

- 2** Continue installing the kernel. Type

```
y
```

MATLAB installs the kernel and displays the message

```
The Real-Time Windows Target kernel has been successfully  
installed.
```

If a message is displayed asking you to restart your computer, you need to restart your computer before the kernel runs correctly.

- 3** Check that the kernel was correctly installed. Type

```
rtwho
```

MATLAB should display a message that shows the kernel version number, followed by performance, timeslice, and other information.

Once the kernel is installed, you can leave it installed. After you have installed the kernel, it remains idle, which allows Windows to control the execution of any standard Windows application. Standard Windows applications include internet browsers, word processors, MATLAB, and so on.

It is only during real-time execution of your model that the kernel intervenes to ensure that your model is given priority to use the CPU to execute each model update at the prescribed sample intervals. Once the model update at a particular sample interval completes, the kernel releases the CPU to run any other Windows application that might need servicing.

Uninstalling the Kernel

If you encounter any problems with Real-Time Windows Target, you can uninstall the kernel. The kernel executable file remains on your hard drive so that you can reinstall it:

- 1** In the MATLAB window, type

```
rtwintgt -uninstall
```

MATLAB displays the message

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

- 2** To continue uninstalling the kernel, type

```
y
```

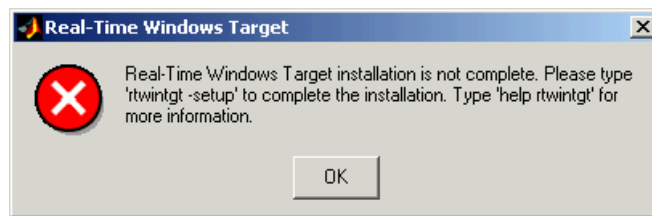
MATLAB uninstalls the kernel by removing it from memory and displays the message

```
The Real-Time Windows Target kernel has been successfully
uninstalled.
```

3 To check that the kernel was correctly uninstalled, type

```
rtwho
```

MATLAB should display the following message.



Once uninstalled, the kernel is no longer active, and has no impact on the operation of your computer.

There are two additional ways to uninstall the Real-Time Windows Target kernel. They are useful if you uninstall MATLAB before you uninstall the kernel.

To uninstall the kernel, click the MATLAB **Start** button, and select **Simulink > Real-Time Windows Target > Uninstall real-time kernel**.

Alternately, from the DOS prompt of your computer, type

```
rtwintgt -uninstall
```

and the kernel will uninstall from your system. Typing

```
rtwintgt -forceuninstall
```

forcibly deregisters the kernel from the operating system without deleting any files. This option should only be used when all other attempts to uninstall

the kernel fail. This command can be used both within MATLAB and at the DOS prompt.

Testing the Installation

- “Running the Model `rtvdp.mdl`” on page 2-14
- “Displaying Status Information” on page 2-16
- “Detecting Excessive Sample Rates” on page 2-17
- “Demo Library” on page 2-18

Real-Time Windows Target includes several demo models. You can use one of the demo models to test your installation. Demo models simplify testing of your installation since they are configured with settings that include the correct target, scope settings, sample time, and integration algorithm.

Once you have completed the installation of Real-Time Windows Target and the kernel, we recommend a quick test by at least running the model `rtvdp.mdl`. If you change your installation, we also recommend doing this test as a quick check to confirm that Real-Time Windows Target is still working.

Running the Model `rtvdp.mdl`

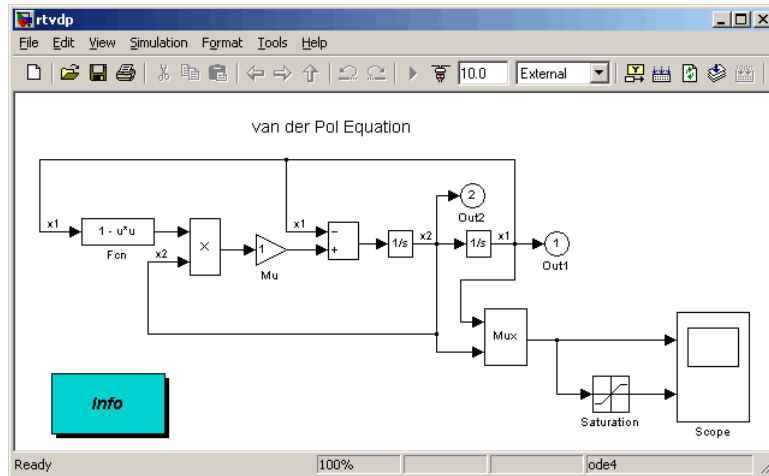
The model `rtvdp.mdl` does not have any I/O blocks, so that you can run this model regardless of the I/O boards in your computer. Running this model will test the installation by running Real-Time Workshop, Real-Time Windows Target, and the Real-Time Windows Target kernel.

After you have installed the Real-Time Windows Target kernel, you can test the entire installation by building and running a real-time application. Real-Time Windows Target includes the model `rtvdp.mdl`, which already has the correct Real-Time Workshop options selected for you:

- 1 In the MATLAB window, type

```
rtvdp
```

The Simulink model `rtvdp.mdl` window opens.



2 From the **Tools** menu, choose **Real-Time Workshop > Build Model**.

The MATLAB window displays the following messages:

```

### Starting Real-Time Workshop build for model: rtvdp
### Invoking Target Language Compiler on rtvdp.rtw
. . .
### Compiling rtvdp.c
. . .
### Created Real-Time Windows Target module rtvdp.rwd.
### Successful completion of Real-Time Workshop build procedure
for model: rtvdp

```

3 From the **Simulation** menu, click **External**, and then click **Connect to target**.

The MATLAB window displays the following message:

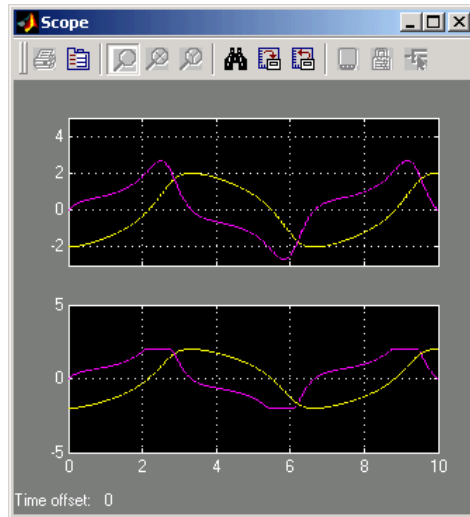
```

Model rtvdp loaded

```

4 From **Simulation** menu, click **Start Real-Time Code**.

The Scope window displays the output signals. If your Scope window looks like the next figure, you have successfully installed Real-Time Windows Target and have run a real-time application.



5 From **Simulation** menu, click **Stop Real-Time Code**.

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

Displaying Status Information

Real-Time Windows Target provides the command `rtwho` for displaying the kernel version number, followed by performance, timeslice, 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:

```
MATLAB performance = 100.0%
```

This message indicates that MATLAB and other non-real-time applications (for example, a word processor) are able to run at 100% performance because no real-time applications are currently executing. When a real-time application is executing, the MATLAB performance is at a value below 100%. For example, if the MATLAB performance = 90.0%, then the real-time application is using 10% of the CPU time. We recommend that you select a sample rate so that `rtwho` returns a MATLAB performance of at least 80%.

```
Kernel timeslice period = 1 ms
```

The kernel time slice period is the current frequency of the hardware timer interrupt. One millisecond is the maximum value for models with large sample times (slow sampling rate) or when an application has not been built. This value changes when you select sampling times less than 1 millisecond.

```
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.

Detecting Excessive Sample Rates

If your specified sample rate is too fast, Real-Time Windows Target detects and reports this during real-time execution. Sampling rates exceeding 10 kHz can be achieved on Pentium computers. Once the model is running, you can issue the `rtwho` command in the MATLAB Command Window to observe the system performance.

For example, the following lines show that MATLAB performance has decreased because the system is overloaded:

```
MATLAB performance = 77.1%  
Kernel timeslice period = 0.001 ms
```

We recommend that MATLAB performance not fall below 80%.

Demo Library

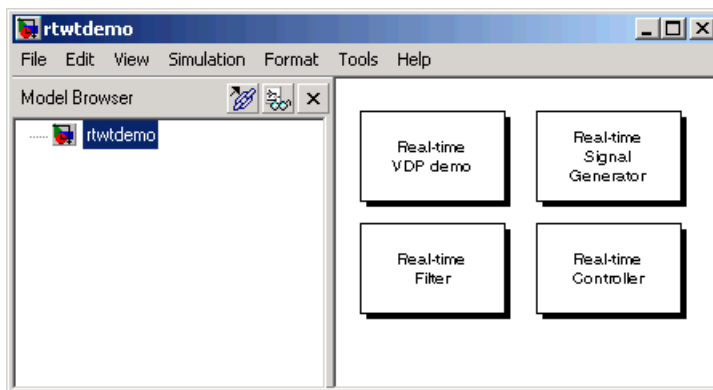
The demo library includes models with preset values and dialog boxes. These models include a configuration of examples that use no I/O, A/D only, A/D and D/A in a simple signal processing demo, as well as in a simple control demo.

Examples that use I/O blocks require you to configure the Adapter block to match the I/O board installed in your computer:

- 1 In the MATLAB window, type

```
rtwtdemo
```

The **rtwtdemo** window opens and displays the demo models provided with Real-Time Windows Target.



- 2 Double-click a demo block to open the model.

Basic Procedures

The basic procedures explain how to create a Simulink or real-time application, and how to run a simulation or execution.

Simulink Model (p. 3-2)

Create a Simulink model and run a non-real-time simulation

Real-Time Application (p. 3-13)

Create a real-time application, generate code from that model, and run a real-time execution

Signal Logging to the MATLAB Workspace (p. 3-26)

Save data from a simulation or execution, and then analyze or visualize that data

Signal Logging to a Disk Drive (p. 3-33)

Save data from a real-time execution, and then analyze or visualize that data

Parameter Tuning (p. 3-43)

Change parameters in your application while it is running in real time

Simulink Model

- “Creating a Simulink Model” on page 3-2
- “Entering Configuration Parameters for Simulink” on page 3-6
- “Entering Scope Parameters for Signal Tracing” on page 3-7
- “Running a Non-Real-Time Simulation” on page 3-10
- “Specifying a Default Real-Time Windows Target Configuration Set” on page 3-11

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.

Creating a Simulink Model

This procedure explains how to create a simple Simulink model. You use this model as an example to learn other procedures in Real-Time Windows Target.

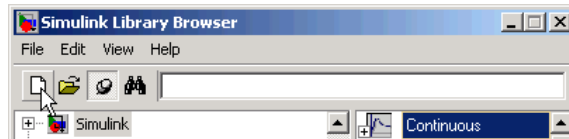
You need to create a Simulink model before you can run a simulation or create a real-time application:

- 1 In the MATLAB Command Window, type

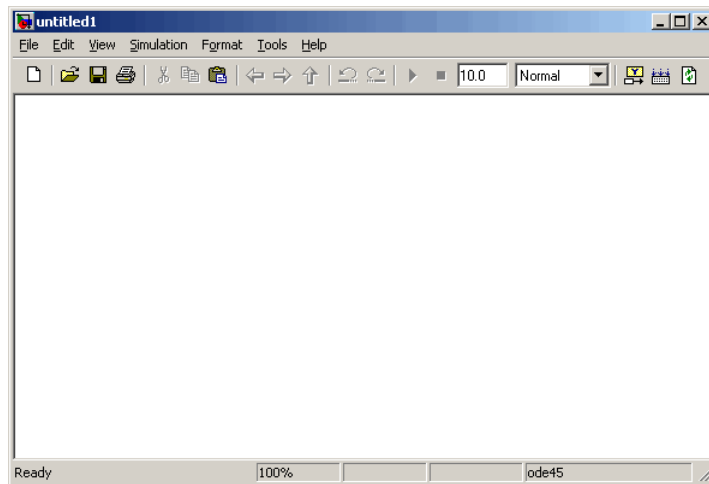
```
simulink
```

The Simulink Library Browser window opens.

- 2 From the toolbar, click the **Create a new model** button.



An empty Simulink window opens:



- 3 In the Simulink Library Browser window, double-click **Simulink**, and then double-click **Sources**. Click and drag **Signal Generator** to the Simulink window.

Click **Continuous**. Click and drag **Transfer Fcn** to the Simulink window.

Click **Sinks**. Click and drag **Scope** to the Simulink window.

- 4 Connect the **Signal Generator** output to the **Transfer Fcn** input by clicking-and-dragging a line between the blocks. Likewise, connect the **Transfer Fcn** output to the **Scope** input.

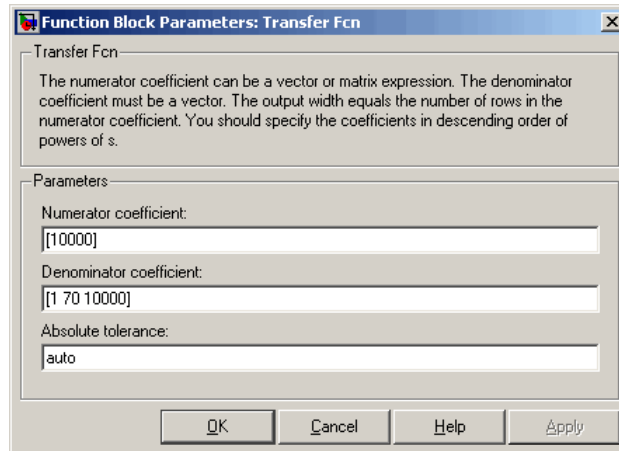
- 5 Double-click the **Transfer Fcn** block. The Block Parameters dialog box opens. In the **Numerator** text box, enter

[10000]

In the **Denominator** text box, enter

[1 70 10000]

Your Block Parameters dialog box looks similar to the next figure.



6 Click **OK**.

7 Double-click the Signal Generator block. The Block Parameters dialog box opens. From the **Wave form** list, select square.

In the **Amplitude** text box, enter

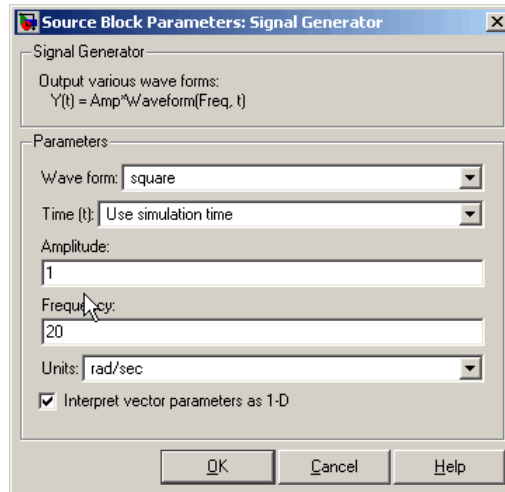
1

In the **Frequency** text box, enter

20

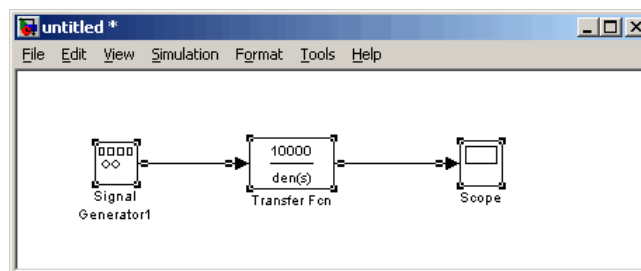
From the **Units** list, select rad/sec.

Your Block Parameters dialog box looks similar to the next figure.



8 Click **OK**.

The next figure shows the completed Simulink block diagram, with toolbar and status bar not shown:



9 From the **File** menu, click **Save As**. The Save As dialog box opens. In the **File name** text box, enter a filename for your Simulink model and click **Save**. For example, type

```
rtwin_model
```

Simulink saves your model in the file `rtwin_model.mdl`.

To specify a default Real-Time Windows Target configuration set for your model, see “Specifying a Default Real-Time Windows Target Configuration Set” on page 3-11. If you activate this configuration set for your model, you can build your real-time application later without setting additional configuration parameters.

To manually configure your model, continue to “Entering Configuration Parameters for Simulink” on page 3-6, following. That section teaches you how to enter configuration parameters for your Simulink model, then leads you into procedures for entering scope parameters and running a non-real-time simulation of the model.

Entering Configuration Parameters for Simulink

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

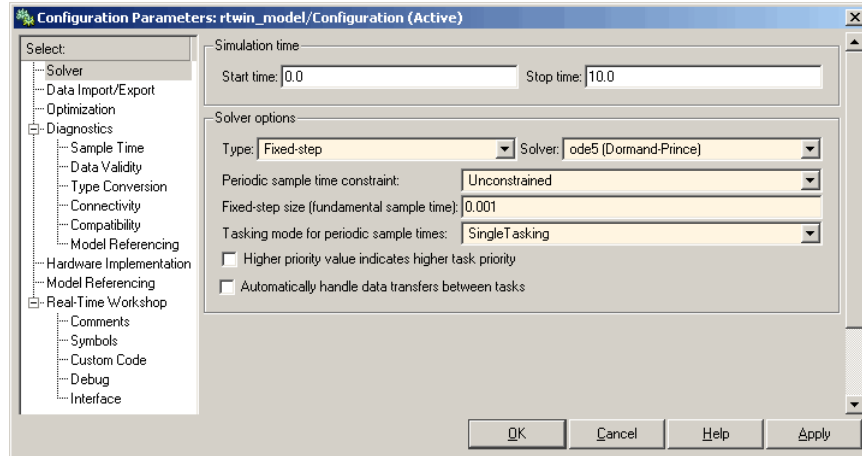
After you create a Simulink model, you can enter the configuration parameters for Simulink. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model:

- 1** In the Simulink window, and from the **Simulation** menu, click **Configuration Parameters**. In the Configuration Parameters dialog box, click the **Solver** tab.

The **Solver** pane opens.

- 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. Real-Time Workshop 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** 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** From the **Tasking Mode** list, choose `SingleTasking`. (For models with blocks that have different sample times, choose `MultiTasking`.)

Your **Solver** pane looks similar to the next figure.



7 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the dialog box.

Entering 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.

A Scope window opens.

- 2** Click the **Parameters** button.



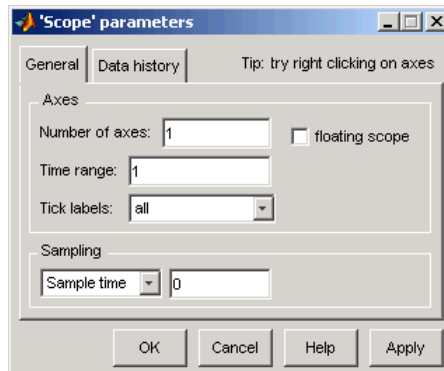
A Scope parameters dialog box opens.

- 3** Click the **General** tab. In the **Number of axes** field, enter the number of graphs you want in one Scope window. For example, enter 1 for a single graph. Do not select the **floating scope** check box.

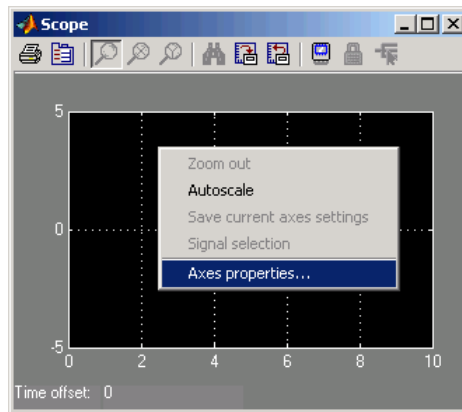
In the **Time range** field, enter the upper value for the time range. For example, enter 1 second. From the **Tick labels** list, choose all.

From the **Sampling** list, choose Sample time and enter 0 in the text box. Entering 0 indicates that Simulink evaluates this block as a continuous time block. If you have discrete blocks in your model, enter the **Fixed step size** you entered in the Configuration Parameters dialog box.

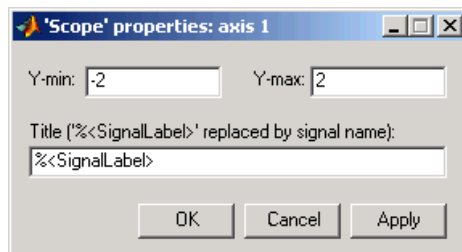
Your Scope parameters dialog box looks similar to the next figure.



- 4 Do one of the following:
 - Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the dialog box.
- 5 In the Scope window, point to the y-axis shown in the next figure, and right-click.



- 6 From the pop-up menu, click **Axes Properties**.
- 7 The Scope properties: axis 1 dialog box opens. 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 as shown in the next figure.



8 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the dialog box.

Running a Non-Real-Time Simulation

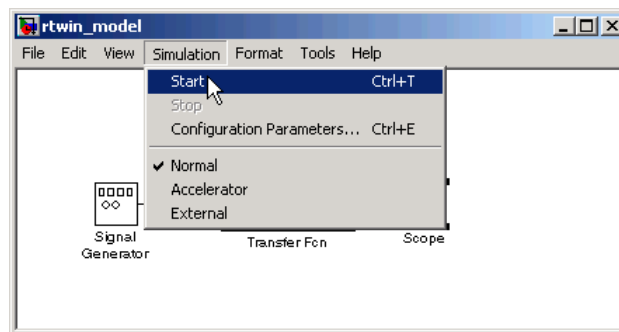
You use Simulink in normal mode to run a non-real-time simulation. Running a simulation lets you observe the behavior of your model in nonreal time.

After you load your Simulink model into the MATLAB workspace, you can run a simulation. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have loaded that model:

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

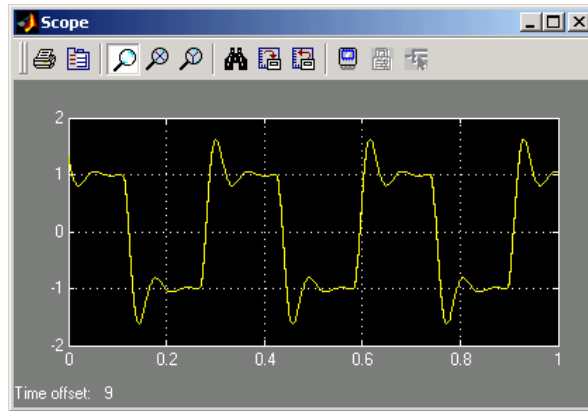
Simulink opens a Scope window with an empty graph.

2 From the **Simulation** menu, click **Normal**, and then click **Start**.



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

During the simulation, the Scope window displays the samples for one time range, increases the time offset, and then displays the samples for the next time range.



3 Do one of the following:

- Let the simulation run to the stop time.
- From the **Simulation** menu, click **Stop**.

The simulation stops. MATLAB does not display any messages.

Specifying a Default Real-Time Windows Target Configuration Set

The preceding sections describe how to use the Simulink Configuration Parameters dialog to configure simulation parameters for a Simulink model. To quickly configure a Simulink model for default Real-Time Windows Target behavior, use the default Real-Time Windows Target configuration set.

A configuration set is a named set of values for model parameters, such as solver type and simulation start or stop time. Every new model is created with a default configuration set, called Configuration, that initially specifies default values for the model's model parameters. You can subsequently create additional configuration sets and associate them with the model. See the Simulink documentation for further details on configuration sets.

After you create a Simulink model, you can specify a default Real-Time Windows Target configuration set for the model. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model (see “Creating a Simulink Model” on page 3-2):

- 1** If you have not already saved the model, from the **File** menu, click **Save As**. The Save As dialog box opens. In the **File name** text box, enter a filename for your Simulink model and click **Save**. For example, type

```
rtwin_model
```

Simulink saves your model in the file `rtwin_model.mdl`.

- 2** In the MATLAB window, type

```
rtwinconfigset('rtwin_model')
```

The default Real-Time Windows Target configuration set, RTWin, is active for the `rtwin_model` model. You do not need to perform any other configuration to build a Real-Time Windows Target application.

- 3** Save the model.

See “Creating a Real-Time Application” on page 3-18 for a description of how to build your Real-Time Windows Target application.

Note To revert to the default configuration set, Configuration, or any other configuration set you have for the model, use Model Explorer. This is an alternative tool that you can use to enter simulation parameters for a model. This document does not describe how to use the Model Explorer. See the Simulink documentation for a description of how to use Model Explorer.

Real-Time Application

- “Entering Simulation Parameters for Real-Time Workshop” on page 3-13
- “Entering Scope Parameters for Signal Tracing” on page 3-16
- “Creating a Real-Time Application” on page 3-18
- “Entering Additional Scope Parameters for Signal Tracing” on page 3-19
- “Running a Real-Time Application” on page 3-21
- “Running a Real-Time Application from the MATLAB Command Line” on page 3-24

You create a real-time application to let your system run while synchronized to a real-time clock. This allows your system to control or interact with an external system. This is necessary if you use your system to stabilize a physical plant.

The process of creating and running a real-time application includes the creation of a Simulink Model from the previous section:

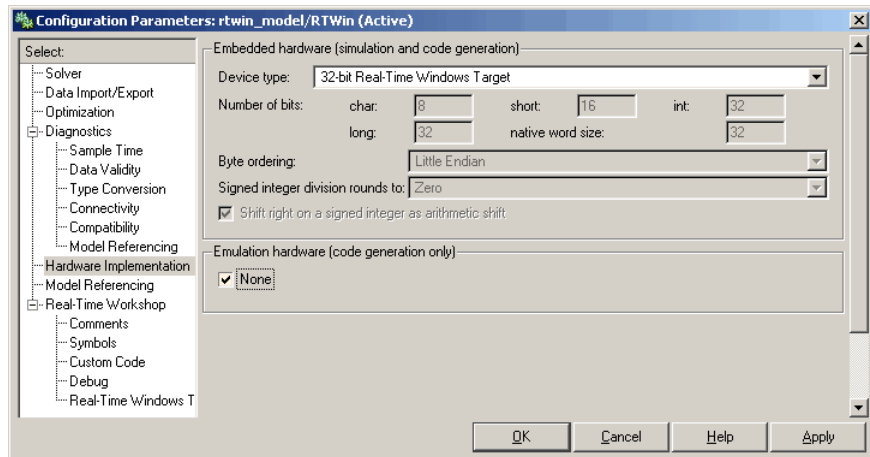
- “Creating a Simulink Model” on page 3-2
- “Entering Configuration Parameters for Simulink” on page 3-6
- “Specifying a Default Real-Time Windows Target Configuration Set” on page 3-11

Entering Simulation Parameters for Real-Time Workshop

After you create a Simulink model, you can enter the simulation parameters for Real-Time Workshop. The simulation parameters are used by Real-Time Workshop for generating C code and building a real-time application.

This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model:

- 1 In the Simulink window, and from the **Simulation** menu, click **Configuration Parameters**.
- 2 Click the **Hardware Implementation** node.
- 3 From the **Device type** list, choose 32-bit Real-Time Windows Target.
- 4 Under **Emulation hardware**, select None.



- 5 Click the **Real-Time Workshop** node.

The **Real-Time Workshop** pane opens.

- 6 In the **Target selection** section, click the **Browse** button at the **RTW system target file** list.

The **System Target File Browser** opens.

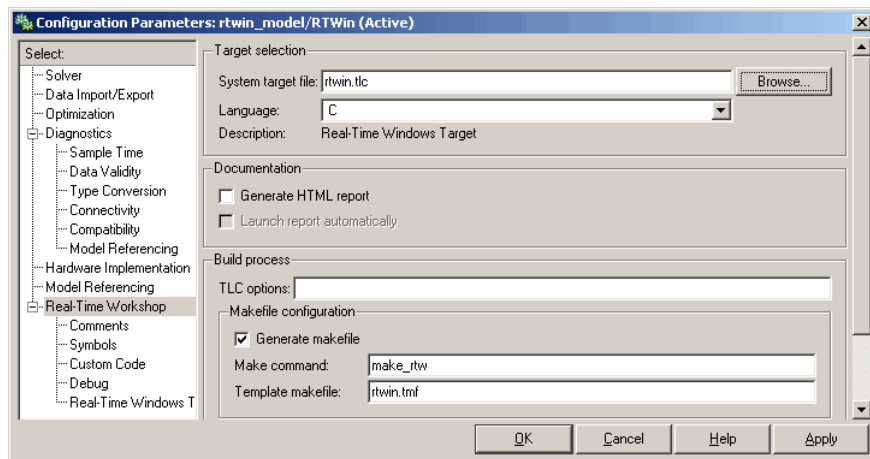
- 7 Select the system target file for Real-Time Windows Target and click **OK**.

rsim.tlc	Rapid Simulation Target
rtwin.tlc	Real-Time Windows Target
rtwsfcn.tlc	S-function Target

The system target file `rtwin.tlc`, the template makefile `rtwin.tmf`, and the make command `make_rtw` are automatically entered into the **Real-Time Workshop** pane.

Although not visible in the **Real-Time Workshop** pane, the external target interface MEX file `rtwinext` is also configured when you click **OK**. This 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.

Your **Real-Time Workshop** pane looks similar to the next figure.



Do not select **Inline parameters** on the **Optimization** node. Inlining parameters is used for custom targets when you want to reduce the amount of RAM or ROM with embedded systems. Also, if you select inlining parameters, the parameter tuning feature is disabled. Since PCs have more memory than embedded systems, we recommend that you do not inline parameters.

8 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the dialog box.

Entering Scope Parameters for Signal Tracing

You enter or change scope parameters to format the x -axis and y -axis in a Scope window. Other parameters include the number of graphs in a one Scope window and whether the scope is connected to a continuous or discrete model.

If you entered the scope parameters for running a simulation, you can skip this procedure. This information is repeated here if you did not run a simulation.

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.

A Scope window opens.

2 Click the Parameters button.



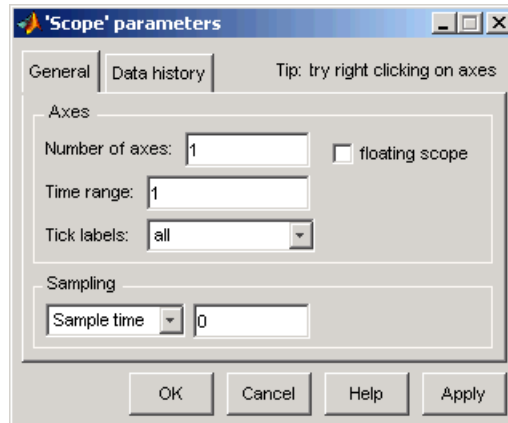
A Scope parameters dialog box opens.

3 Click the **General** tab. In the **Number of axes** field, enter the number of graphs you want in one Scope window. For example, enter 1 for a single graph. Do not select the **floating scope** check box.

In the **Time range** field, enter the upper value for the time range. For example, enter 1 second. From the **Tick labels** list, choose all.

From the **Sampling** list, choose **Sample time** and enter 0 in the text box. Entering 0 indicates that Simulink evaluates this block as a continuous time block. If you have discrete blocks in your model, enter the **Fixed step size** you entered in the Configuration Parameters dialog box.

Your Scope parameters dialog box looks similar to the next figure.



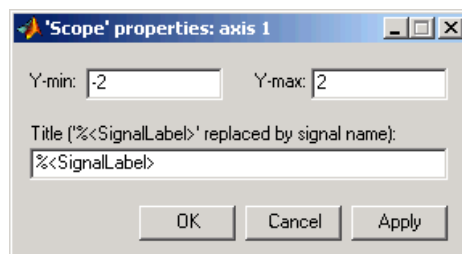
4 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the dialog box.

5 In the Scope window, point to the y-axis and right-click. From the menu, click **Axes Properties**.

The Scope properties: axis 1 dialog box opens.

6 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.



7 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the dialog box.

Creating a Real-Time Application

Real-Time Workshop generates C code from your Simulink model, then the Open Watcom C/C++ compiler compiles and links that C code into a real-time application.

After you enter parameters into the Configuration Parameters dialog box for Real-Time Workshop, you can build a real-time application. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you have loaded that model:

1 In the Simulink window, and from the **Tools** menu, point to **Real-Time Workshop**, and then click **Build Model**.

The build process does the following:

- Real-Time Workshop creates the C code source files `rtwin_model.c` and `rtwin_model.h`.
- The make utility `make_rtw.exe` creates the makefile `rtwin_model.mk` from the template makefile `rtwin.tmf`.
- The make utility `make_rtw.exe` builds the real-time application `rtwin_model.rwd` using the makefile `rtwin_model.mk` created above. The file `rtwin_model.rwd` is a binary file that we refer to as your real-time application. You can run the real-time application with the Real-Time Windows Target kernel.

2 Connect your Simulink model to your real-time application. See “Entering Additional Scope Parameters for Signal Tracing” on page 3-19.

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.

Entering Additional Scope Parameters for Signal Tracing

Simulink external mode connects your Simulink model to your real-time application. This connection allows you to use the Simulink block diagram as a graphical user interface to your real-time application.

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, and from the Simulation menu, click **Configuration Parameters**.

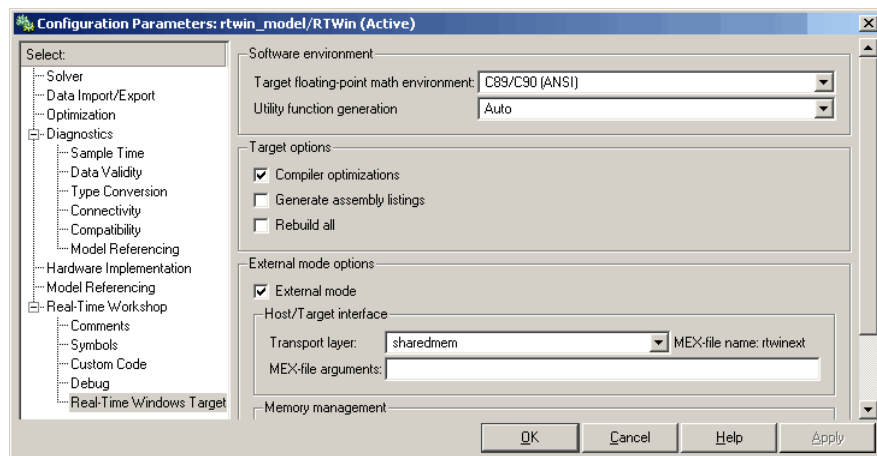
2 Select the **Real-Time Windows Target** node.

The **Real-Time Windows Target** pane opens.

3 Select the **External mode** check box.

The **MEX-file name** label should have an entry of `rtwinext`. The MEX-file `rtwinext.mex*` is supplied with Real-Time Windows Target to work with Simulink external mode and support uploading signal data and downloading parameter values.

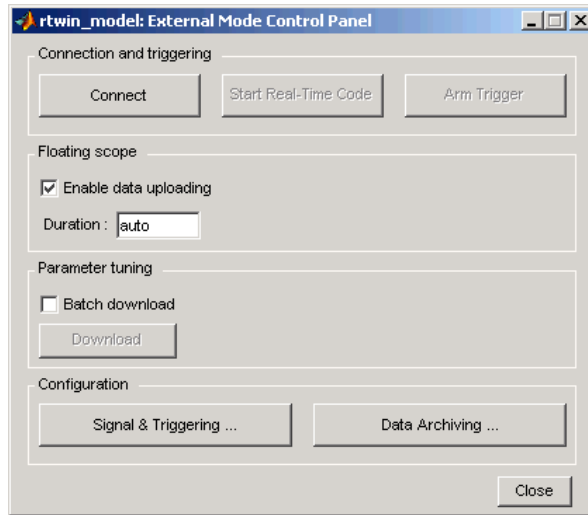
The **Real-Time Windows Target** pane should appear as follows.



4 Click **OK**.

5 In the Simulation window, and from the **Tools** menu, click **External Mode Control Panel**.

The External Mode Control Panel dialog box opens.



6 Click the **Signal & Triggering** button.

The External Signal & Triggering dialog box opens.

7 Select the **Select all** check box. From the **Source** list, choose **manual**. From the **Mode** list, choose **normal**.

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

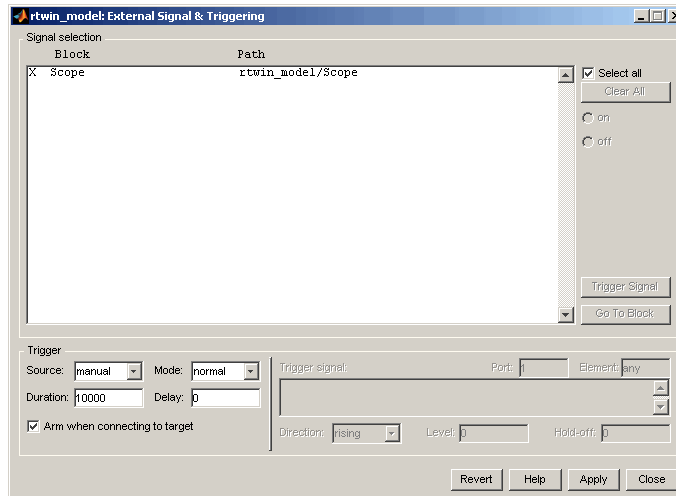
8 In the **Duration** field, enter the number of sample points in a data buffer. For example, to specify a sample rate of 1000 samples/second and a stop time of 10 seconds, enter

10000

9 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.

The External Signal & Triggering dialog box looks like this:



10 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **Close** to apply the changes to your model and close the dialog box.


You must click the **Apply** or **Close** button on the External Signal & Triggering dialog box for the changes you made to take effect. Generally it is not necessary to rebuild your real-time application.

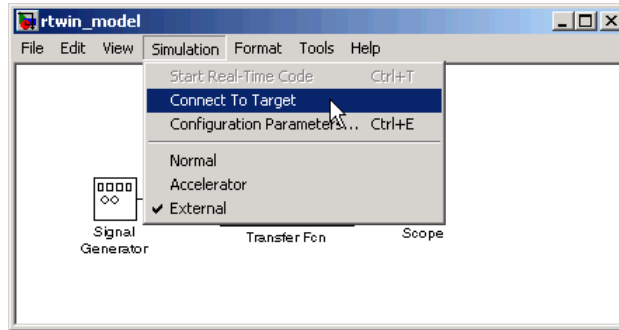
Running a Real-Time Application

You run your real-time application to observe the behavior of your model in real time with the generated code. The process of connecting consists of

- Establishing a connection between your Simulink model and the kernel to allow exchange of commands, parameters, and logged data.
- Running the application in real time.


After you build the real-time application, you can run your model in real time. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you have created a real-time application for that model:

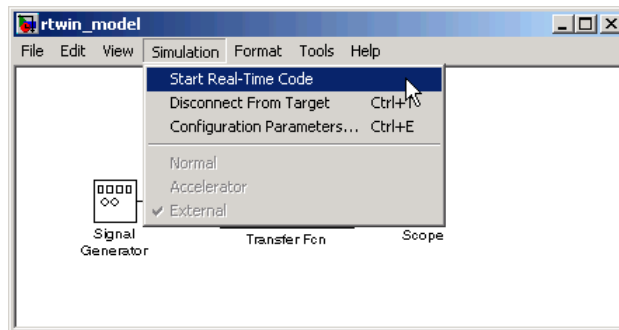
- 1 From the **Simulation** menu, click **External**, then from the **Simulation** menu click **Connect To Target**. Also, you can connect to the target from the toolbar by clicking .



MATLAB displays the message

Model `rtwin_model` loaded

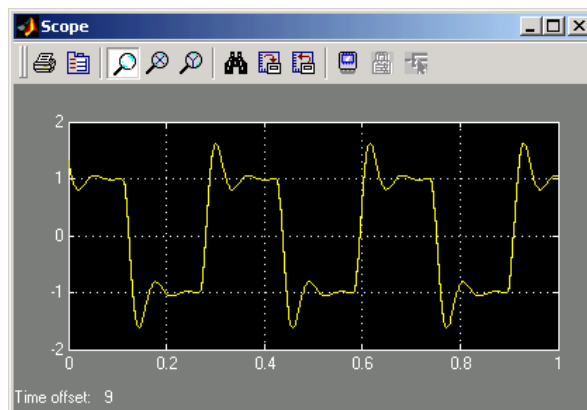
- 2 In the Simulation window, and from the **Simulation** menu, click **Start Real-Time Code**. You can also start the execution from the toolbar by clicking .



Simulink runs the execution and plots the signal data in the Scope window.

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.

Note 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 remaining CPU time after real-time application computations are performed while waiting for another interrupt to trigger the next real-time application update. The result may be a loss of data points displayed in the Scope window.



3 Do one of the following:

- Let the execution run until it reaches the stop time.
- From the **Simulation** menu, click **Stop Real-time Code**.

The real-time application stops.

4 In the Simulation window, and from the **Simulation** menu, click **Disconnect From Target**.

- 5** From the **Simulation** menu, click **External**.

MATLAB displays the message

```
Model rtwin_model unloaded
```

Running a Real-Time Application from the MATLAB Command Line

You can use the MATLAB command-line interface as an alternative to using the Simulink GUI. Enter commands directly in the MATLAB window or enter them in an M-file.

After you build the real-time application, you can run your model in real time. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you have created a real-time application for that model:

- 1** In the MATLAB window, type

```
set_param(gcs, 'SimulationMode', 'external')
```

Simulink changes to external mode.

- 2** Type

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

MATLAB loads the real-time application, connects it to the Simulink block diagram, and displays the message

```
Model rtwin_model loaded
```

- 3** Type

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

Simulink starts running the real-time application.

4 Type

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

Simulink stops the real-time application.

Signal Logging to the MATLAB Workspace

- “Entering Scope Parameters” on page 3-26
- “Entering Signal and Triggering Properties” on page 3-28
- “Plotting Logged Signal Data” on page 3-31

Signal logging is the process of saving (logging) data to a variable in your MATLAB workspace or to a MAT-file on your disk drive. This allows you to 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.

To use signal logging with Real-Time Windows Target, you must add a Scope block to your Simulink model.

Simulink external mode does not support data logging with Outport blocks in your Simulink model. This means you do not enter or select parameters on the **Data I/O** tab in the Configuration Parameters dialog box.

Entering Scope Parameters

Data is saved to the MATLAB workspace through a Simulink Scope block. Scope block parameters need to be set for data to be saved.

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 Simulink model `rtwin_model.mdl` as an example, and assumes you have already loaded that model.

Note If you entered the scope parameters for running a simulation, you may want to look over this procedure, because the Scope parameters dialog box is related to the External Signal and Triggering dialog box.

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

A Scope window opens.

- 2 On the toolbar, click **Parameters**.



A Scope Parameters dialog box opens.

- 3 Click the **Data history** tab.

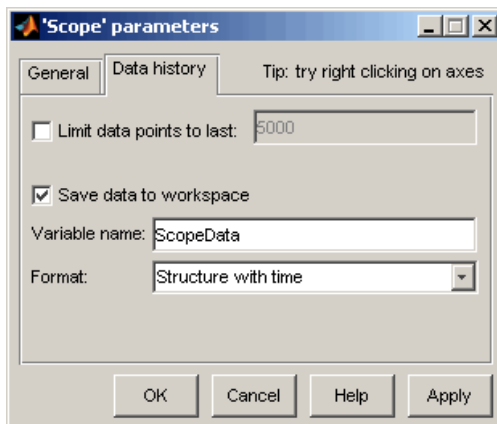
- 4 Do one of the following:

- 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.

The **Limit data points to last** check box is related to the **Duration** value in the External Signal and Triggering dialog box. The smaller of either value limits the number of sample points saved to the MATLAB workspace. When you are using Real-Time Windows Target, use the **Duration** value to set the number of sample points you save. To set the **Duration** value, see the next section.

- 5 Select the **Save 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 Format list, choose Structure with time, Structure, or Array (compatible with V2.0-2.2). For example, to save the sample times and signal values at those times, choose Structure with time.

Your **Data history** pane looks similar to the next figure.



7 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the dialog box.

When you modify anything in the Scope parameters dialog box, you must click the **Apply** or **OK** button for the changes to take effect, and you must rebuild your real-time application before connecting and starting it. If you do not rebuild, an error dialog box will open. If you do not click **Apply**, your executable will run, but it will use the old settings.

The reason why you need to rebuild is because the model checksum includes settings from the Scope block used for signal logging. If the model checksum does not match the checksum in the generated code, the real-time application cannot run. Always rebuild your real-time application after changing Scope parameters.

Entering Signal and Triggering Properties

Data is saved to the MATLAB workspace through a Simulink Scope block. Signal and triggering properties need to be set only when you are running a real-time application. If you are running a 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 Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model:

- 1** In the Simulink window, and from the **Tools** menu, click **External Mode Control Panel**.

The External Mode Control Panel dialog box opens.

- 2** Click the **Signal & Triggering** button.

The External Signal & Triggering dialog box opens.

- 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. For example, if you have a sample rate of 1000 samples/second and a stop time of 10 seconds, you could enter

10000

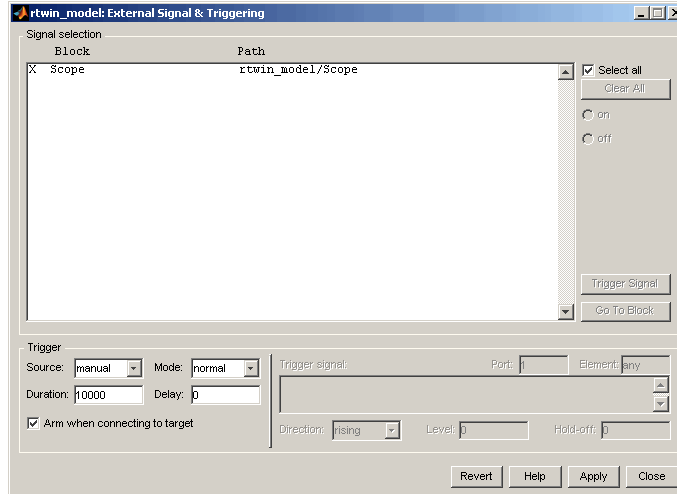
The **Duration** value is related to the **Limit data points to last** value in the Scope parameters dialog box. The smaller of either value limits the number of sample points saved to the MATLAB workspace. We recommend that you do not select the **Limit data points to last** check box; use the **Duration** value to set the number of sample points saved. To clear the **Limit data points to last** check box, see “Entering Scope Parameters” on page 3-26.

The **Duration** value specifies the number of contiguous points of data to be collected in each buffer of data. We recommend that you enter a **Duration** value equal to the total number of sample points that you need to collect rather than relying on a series of buffers to be continuous.

If you enter a value less than the total number of sample points, you will lose sample points during the time needed to transfer values from the data buffer to the MATLAB workspace. Real-Time Windows Target ensures that points are continuous only within one buffer. Between buffers, because of transfer time, some samples will be omitted.

We also recommend setting 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 will display one buffer of data across the entire Simulink Scope plot.

The External Signal & Triggering dialog box looks similar to the next figure.



5 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **Close** to apply the changes to your model and close the dialog box.

You must click the **Apply** or **Close** button on the External Signal & Triggering dialog box for the changes you made to take effect. Generally it is not necessary to rebuild your real-time application.

Plotting Logged Signal Data

You can use the MATLAB plotting functions for visualizing non-real-time simulated data or real-time application data.

After running your real-time application and logging data to the MATLAB workspace, you can plot the data. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you saved your data to the variable `ScopeData`.

- 1 In the MATLAB window, type

```
ScopeData
```

MATLAB lists the structure of the variable `ScopeData`. The variable `ScopeData` is a MATLAB structure containing fields for the time vector, signal structure, and a string containing the block name.

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

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

```
ScopeData.signals
```

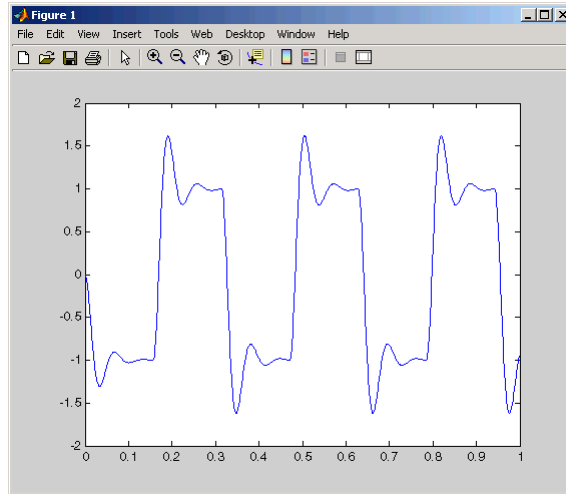
MATLAB lists the structure of the variable `ScopeData.signals`. This structure contains one or more vectors of signal data depending on the number of signal inputs to the Scope block.

```
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))
```

MATLAB plots the first 1000 samples from 0.0000 to 0.9990 second.



3 The variable ScopeData is not automatically saved to your hard disk. To save the variable ScopeData, type

```
save ScopeData
```

MATLAB saves the scope data to the file ScopeData.mat.

Signal Logging to a Disk Drive

- “Entering Scope Parameters” on page 3-33
- “Entering Signal and Triggering Properties” on page 3-36
- “Entering Data Archiving Parameters” on page 3-38
- “Plotting Logged Signal Data” on page 3-40

Signal logging is the process of saving (logging) data to a variable in your MATLAB workspace and then saving that data to a MAT-file on your disk drive. This allows you to use MATLAB functions for data analysis and MATLAB plotting functions for visualization. Using the data archiving feature to provide in the **External Mode Control Panel**, you can save data to a file during execution. You cannot save data to a disk drive during a simulation.

To use the data archiving feature with Real-Time Windows Target, you must add a Scope block to your Simulink model, and you must execute a real-time application.

Simulink external mode does not support data logging with Outport blocks in your Simulink model. This means you do not enter or select parameters on the **Data I/O** pane in the Configuration Parameters dialog box.

Entering Scope Parameters

You save data to a disk drive by first saving the data to the MATLAB workspace through a Simulink Scope block. You need to set scope block parameters for data to be saved.

After you create a Simulink model and add a Scope block, you can enter the scope parameters for signal logging to a disk drive. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you have already loaded that model.

Note If you entered the scope parameters for running a simulation, you may want to look over this procedure, because the Scope parameters dialog box is related to the External Signal & Triggering dialog box and the Data Archiving dialog box.

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

A Scope window opens.

2 On the toolbar, click the Parameters button.



A Scope parameters dialog box opens.

3 Click the **Data history** tab.

4 Do one of the following:

- If you are running a simulation, you can 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.

The **Limit data points to last** check box is related to the **Duration** value in the External Signal & Triggering dialog box. The smaller of either value limits the number of sample points saved to the MATLAB workspace. When using Real-Time Windows Target, we recommend that you use the **Duration** value to set the number of sample points you save. To set the **Duration** value, see “Entering Signal and Triggering Properties” on page 3-36.

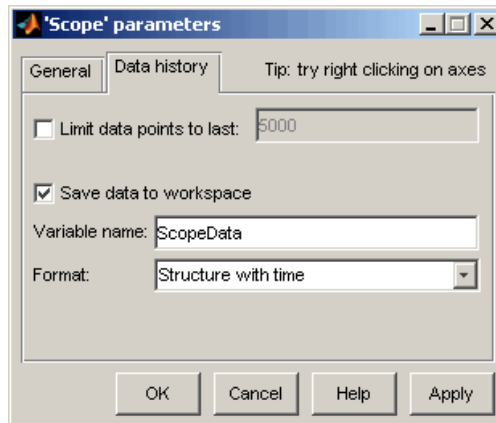
5 Select the **Save data to workspace** check box. In the **Variable name** text box, enter the name of a MATLAB variable. The default name is ScopeData.

The Scope parameters dialog box is related to the Data Archiving dialog box. In the Scope parameters dialog box, you must select the **Save data to workspace** check box to be able to save data to a disk drive, for two reasons:

- The data is first transferred from the data buffer to the MATLAB workspace before being written to a MAT-file.
- The **Variable name** entered in the Scope parameters dialog box is the same as the one in the MATLAB workspace and the MAT-file.

If you do not select the **Save data to workspace** check box, the MAT-files for data logging will be created, but they will be empty.

- 6 From the Format list, choose either Structure with time, Structure, or Array (compatible with Version 2.0 through Version 2.2). For example, to save the sample times and signal values at those times, choose Structure with time. Your **Data history** pane looks similar to the next figure.



- 7 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the dialog box.

You must rebuild your real-time application before connecting and starting the application with changed settings. If you do not rebuild after making changes, an error will occur.

Entering Signal and Triggering Properties

Data is saved to a disk drive by first saving the data to the MATLAB workspace through a Simulink Scope block. Signal and triggering properties need to be set when running a real-time application.

After you create a Simulink model and add a Scope block, you can enter the signal and triggering properties for data logging to a disk drive. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you have already loaded that model:

- 1** In the Simulink window, and from the **Tools** menu, click **External Mode Control Panel**.

The External Mode Control Panel dialog box opens.

- 2** Click the **Signal & Triggering** button.

The External Signal & Triggering dialog box opens.

- 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. For example, if you have a sample rate of 1000 samples/second and a stop time of 10 seconds, then enter

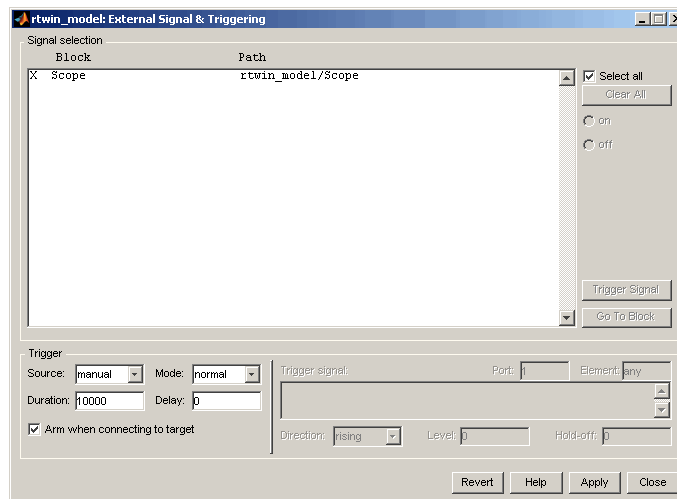
10000

The **Duration** value is related to the **Limit data points to last** value in the Scope parameters dialog box. The smaller of either value limits the number of sample points saved to the MATLAB workspace. We recommend that you do not select the **Limit data points to last** check box, and use the **Duration** value to set the number of sample points saved.

The **Duration** value specifies the number of contiguous points of data to be collected in each buffer of data. We recommend that you enter a **Duration** value equal to the total number of sample points you need to collect for a run. If you enter a value much less than the total number of sample points, you may lose logging sample points due to the time needed to transfer values from the data buffer to the MATLAB workspace.

We also recommend setting 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 will display one buffer of data across the entire Simulink Scope plot.

The External Signal & Triggering dialog box looks similar to the next figure.



5 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **Close** to apply the changes to your model and close the dialog box.

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

Entering Data Archiving Parameters

The Data Archiving dialog box is related to the Scope parameters dialog box. In the Scope parameters dialog box, you must select the **Save data to workspace** check box to be able to save data to a disk drive, for two reasons:

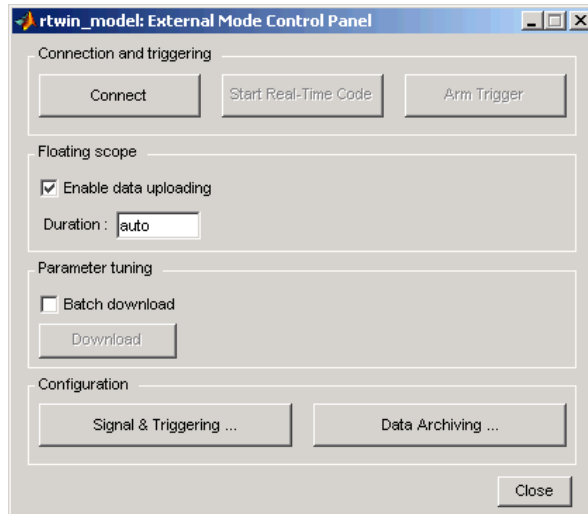
- The data is first transferred from the scope data buffer to the MATLAB workspace before being written to a MAT-file.
- The **Variable name** entered in the Scope parameters dialog box is the same as the one in the MATLAB workspace and the MAT-file. Enabling the data to be saved enables a variable named with the **Variable name** parameter to be saved to a MAT-file.

If you do not select the **Save data to workspace** check box in the Scope parameters dialog box, the MAT-files for data logging will be created, but they will be empty.

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

- 1** In the Simulation window, and from the **Tools** menu, click **External Mode Control Panel**.

The External Mode Control Panel dialog box opens.



- 2** Click the **Data Archiving** button.

The External Data Archiving dialog box opens. This dialog box allows you to specify data archiving options.

- 3** Select the **Enable archiving** check box.

- 4** In the **Directory** text box, enter the path to a directory on your disk drive. For example, if your MATLAB working directory is named `mwd`, enter

```
c:\mwd
```

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

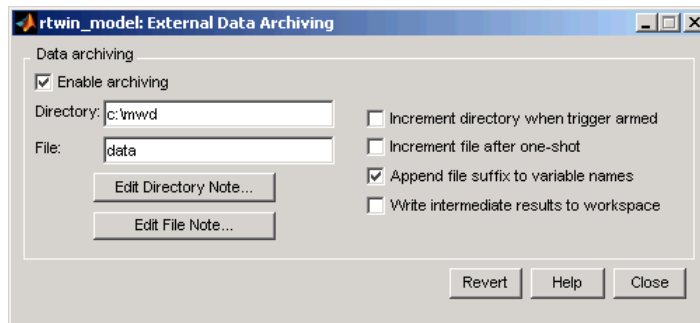
```
data
```

MATLAB names the files `data_0.mat`, `data_1.mat`, etc. 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.

Within each MAT-file, a variable is saved with the same name you entered in the **Variable name** text box (**Data history** pane on the Scope parameters dialog box). By selecting the **Append file suffix to variable names** check box, the same suffix that is added to the MAT-file is added to the variable name. For example, if you entered the variable name ScopeData, then within the file data_0.mat will be a variable ScopeData_0.

Your External Data Archiving dialog box looks similar to the next figure.



7 Click the **Close** button.

The parameters you entered are applied to your model.

There is no **Apply** button with this dialog box. You must click the **Close** button for the changes you make to take effect.

Plotting Logged Signal Data

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 disk drive, you can plot the data. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you saved your data to the variable `ScopeData`:

1 In the MATLAB window, type

```
ScopeData
```

MATLAB lists the structure of the variable ScopeData. The variable ScopeData is a MATLAB structure containing the fields time vector, signal structure, and a string containing the block name.

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

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

```
dir *.mat
```

MATLAB displays the MAT-files in your current working directory.

```
ScopeData.mat
```

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

```
clear  
load ScopeData  
who
```

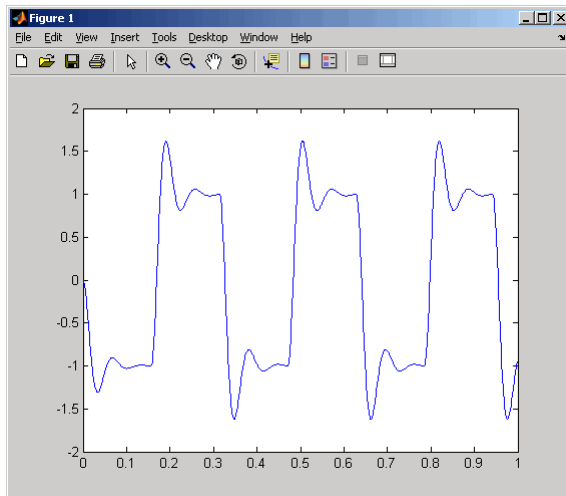
MATLAB displays

```
Your variables are:  
ScopeData
```

4 To plot the first 1000 points, type

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

MATLAB plots the first 1000 samples from 0.0000 to 0.9990 second.



Parameter Tuning

- “Types of Parameters” on page 3-43
- “Changing Model Parameters” on page 3-44

Simulink external mode connects your Simulink model to your real-time application. The block diagram becomes a graphical user interface to the real-time application.

Types of Parameters

You can change parameter values while running the real-time application by changing the values in

- **Block parameters** — Change block parameters by changing the values in the dialog boxes associated with the Simulink blocks. Once you change a value, and click **OK**, the new value is downloaded to the real-time application.

Note Opening a dialog box for a source block causes Simulink to pause. While Simulink is paused, you can edit the parameter values. You must close the dialog box to have the changes take effect and allow Simulink to continue.

- **MATLAB variables** — Change MATLAB variables by entering the changes through the MATLAB command line, and then press **Ctrl+D** for the changes to be downloaded to your executable. An alternative method to download parameters is to click **Update Diagram** from the **Edit** menu in your Simulink window. Simply changing the value of the MATLAB variable at the MATLAB command line is not sufficient for Simulink to know that the value has changed.

Simulink external mode also supports side-effects functions. For example, given an expression in a Gain block of $2*a+b$, the expression is evaluated and the resulting value is exported to the real-time application during execution.

When a parameter in a Simulink model is changed, the communication module `rtwinext.mex*` transfers the data to the external real-time application and changes the model parameters. Only the parameters that do not result in model structure modification can be changed. If the structure is modified, you must recompile the model. Model structure changes are detected automatically using model checksum and reported to the MATLAB window to avoid conflicts.

Changing Model Parameters

You must use Simulink external mode to change model parameters. While external mode is running, you can open Simulink blocks and change parameter values. External mode will transfer the new value to the real-time application.

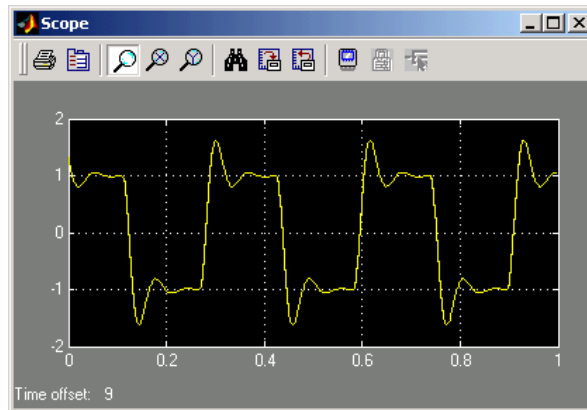
Note Opening a dialog box for a source block causes Simulink to pause. While Simulink is paused, you can edit the parameter values. You must close the dialog box to have the changes take effect and allow Simulink to continue.

After you start running your real-time application, you can change parameters and observe the changes to the signals. To start a real-time application, see “Running a Real-Time Application” on page 3-21.

The following procedure uses the Simulink model `rtwin_model.mdl` as an example. It assumes you have created a real-time application and are executing it.

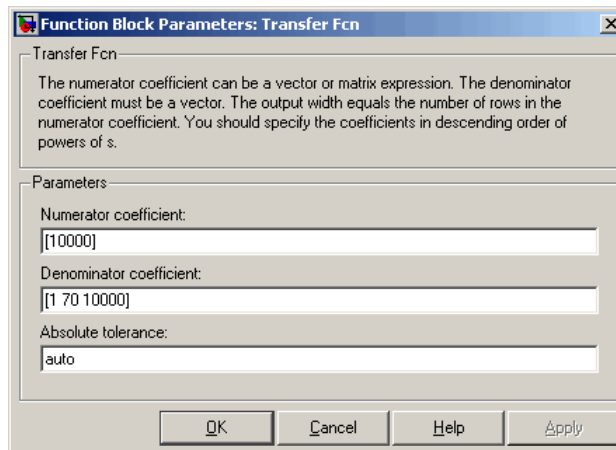
- 1 From the **Simulation** menu, click **Start Real-Time**.

The real-time execution starts running and signal data is displayed in the Scope window.



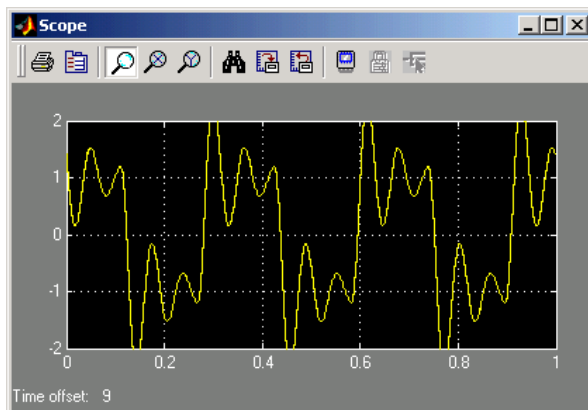
- 2 From the Simulink block diagram, double-click the Transfer Function block.

The Block Parameters: Transfer Fcn dialog box opens.



- 3 In the **Denominator** field, change 70 to 30. Click **OK**.

The effect of changing a block parameter is shown in the Scope window.



Advanced Procedures

Real-Time Windows Target provides driver blocks for more than 200 I/O boards. These driver blocks connect the physical world to your real-time application.

I/O Boards (p. 4-2)

Install I/O boards and enter hardware information

I/O Driver Blocks (p. 4-9)

Select analog and digital driver blocks from the Simulink library and add to your Simulink model

Using Analog I/O Drivers (p. 4-29)

Convert normalized I/O signals to more meaningful model parameters

I/O Boards

- “Installing and Configuring I/O Boards and Drivers” on page 4-2
- “ISA Bus Board” on page 4-6
- “PCI Bus Board” on page 4-7
- “PC/104 Board” on page 4-8
- “Compact PCI Board” on page 4-8
- “PCMCIA Board” on page 4-8

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 any I/O board, read the board manufacturer’s documentation.

For an online list of all I/O boards that Real-Time Windows Target supports, see Supported I/O Boards.

Installing and Configuring I/O Boards and Drivers

A Real-Time Windows Target model connects to a board by including an I/O driver block. This block provides an interface to the board’s device driver and all board-specific settings. The device drivers provided by Real-Time Windows Target 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. This type of setting is necessary when the board manufacturer does not provide a means for the I/O driver to write or read all board settings. Examples include base address, D/A gain, and differential or single-ended A/D inputs.

You can configure a Real-Time Windows Target 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 any jumpers or switches correctly set. Details of installation and configuration depend on the data transfer direction and the specific board, but are essentially similar in all cases. Details for various types of boards and drivers appear later in this chapter.

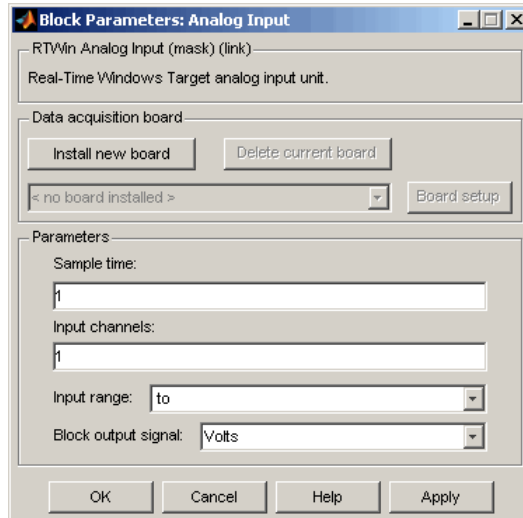
The following instructions use a Humusoft AD512 I/O board as an example, configure the board for analog input, and assume that you have physically configured and installed the board in your computer before you add its driver to your model. Customize the steps to provide the results that you need.

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

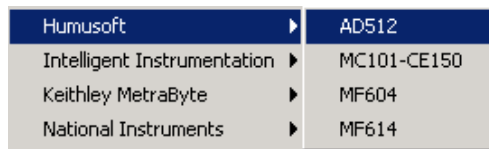
- 1** Install the board in the computer, setting any jumpers or switches as needed.
- 2** Clone an appropriate Input or Output driver block to your model from the Real-Time Windows Target library in the Simulink Library Browser.

3 Double-click the cloned I/O driver block.

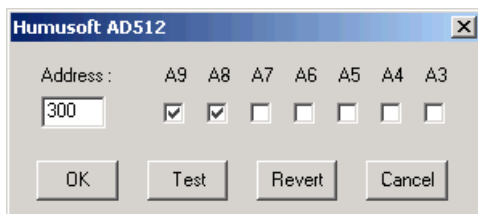
The I/O Block Parameters dialog box opens. For an Analog Input block, the dialog box initially looks like this:



4 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. The box for the Humusoft AD512 board looks like this:



5 Select one of the following, as appropriate to the board:

- 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

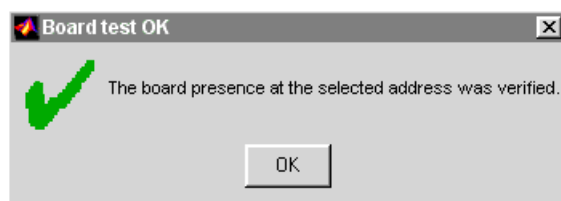
You can also select the base address by checking boxes **A9** through **A3**.

- For a PCI bus board, enter the **PCI slot** or check **Auto-detect**.

6 The I/O Block Parameters dialog also lets you set other I/O block parameters, such as the sample time. Set such parameters as needed.

7 Click **Test**.

Real-Time Windows Target tries to connect to the selected board, and if successful, displays the following message.



- 8 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 would result in two entries in the drop-down list of installed boards. The entries would differ only in the PCI slot number shown for each board.

Autodetecting Multiple Boards. The Autodetect feature cannot be used to locate multiple boards of the same type. You must specify their PCI slot 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 accomplish this, clone the appropriate driver block, open its I/O Block Parameters dialog box, and choose the board from the list of installed boards.

Scope of Driver Block Parameters. All 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 any driver block connected to a board, the same change occurs in all 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 any I/O driver Block Parameters dialog box.

PCI Bus Board

You do not have to set a base address with a PCI board. The plug-and-play feature of Microsoft Windows assigns a PCI slot number. You can enter the slot number in the I/O board dialog box, or you can let the driver determine the slot number for you. You open the I/O board dialog box from any I/O driver Block Parameters dialog box.

We recommend that before you use a PCI or PCMCIA board, you install the drivers supplied by the board manufacturer. Real-Time Windows Target 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 Real-Time Windows Target.

Writing PCI Bus Board Drivers

Real-Time Windows Target applications cannot use Windows DLLs and kernel-mode drivers, which are not suitable for real-time operation. The device drivers supported by Real-Time Windows Target are listed at Supported I/O Boards. 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, which is the technique used by all supported Real-Time Windows Target drivers. If the board registers are I/O-mapped, Real-Time Windows Target supports such programming. If the registers are memory-mapped, programming them would require mapping their memory region to Real-Time Windows Target address space, which Real-Time Windows Target does not support for user-written drivers.

If you want to use an unsupported board that you believe should be supported, or you need assistance with user-written Real-Time Windows device drivers, including the case of a board with memory-mapped registers, contact The MathWorks Technical Support.

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 any I/O driver Block Parameters dialog box.

Compact PCI Board

Using a compact PCI board requires that you use a compact PC (industrial PC). In addition, you need to install Windows, MATLAB, Simulink, and Real-Time Windows Target on the compact PC.

PCMCIA Board

The plug-and-play feature of Microsoft Windows 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 any I/O driver Block Parameters dialog box.

We recommend that before you use a PCI or PCMCIA board, you install the drivers supplied by the board manufacturer. Real-Time Windows Target 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 Real-Time Windows Target.

I/O Driver Blocks

- “Real-Time Windows Target Library” on page 4-9
- “Simulink Library” on page 4-11
- “Analog Input Block” on page 4-12
- “Analog Output Block” on page 4-14
- “Digital Input Block” on page 4-16
- “Digital Output Block” on page 4-17
- “Counter Input Block” on page 4-20
- “Encoder Input Block” on page 4-22
- “Other Input and Other Output Blocks” on page 4-24
- “Output Signals from an I/O Block” on page 4-24
- “Variations with Channel Selection” on page 4-25

The Analog Input, Analog Output, Digital Input, Digital Output, Counter Input, and Encoder Input blocks provide an interface to your physical I/O boards and your real-time application. They ensure that the C code generated with Real-Time Workshop correctly maps block diagram signals to the appropriate I/O channels.

You can have multiple blocks associated with each type of I/O block and board. For example, you can have one Analog Input block for channels 1 to 4 and another block for channels 5 to 8.

For information about writing custom I/O driver blocks to work with Real-Time Windows Target, see Appendix A, “Custom I/O Driver Blocks Reference”.

Real-Time Windows Target Library

This topic describes how to access the Real-Time Windows Target library from the MATLAB Command Window. Real-Time Windows Target 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.

After you create a Simulink model, you can add an I/O block. This procedure adds an Analog Input block and uses the Simulink model `rtwin_model.mdl` as an example:

- 1 In the MATLAB window, type

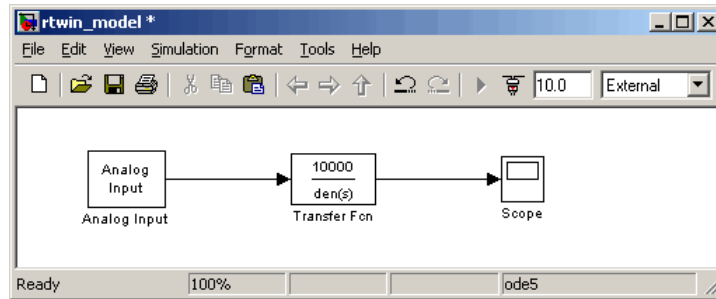
```
rtwinlib
```

The Real-Time Windows Target block library window opens.

Analog Input	Analog Input
Analog Output	Analog Output
Counter Input	Counter Input
Digital Input	Digital Input
Digital Output	Digital Output
Encoder Input	Encoder Input
Other Input	Other Input
Other Output	Other Output

- 2 Click and drag the Analog Input block to your Simulink model. Remove the Signal Generator block and connect the Analog Input block to the Transfer Function block.

Your Simulink model will look similar to the following figure.



Your next task is to enter parameters for the Analog Input block. See “Analog Input Block” on page 4-12.

Simulink Library

This topic describes how to access the Real-Time Windows Target library from the Simulink window. Real-Time Windows Target 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.

After you create a Simulink model, you can add an I/O block. This procedure adds an Analog Input block and uses the Simulink model `rtwin_model.mdl` as an example:

- 1 In the Simulink window, and from the **View** menu, click **Library Browser**.

The Simulink Library Browser opens.

- 2 In the left column, double-click **Real-Time Windows Target**. Click and drag the Analog Input block to your Simulink model. Remove the Signal Generator block and connect the Analog Input block to the Transfer Function block.

Your next task is to enter parameters for the Analog Input block. See “Analog Input Block” on page 4-12.

Analog Input Block

Real-Time Windows Target I/O blocks allow you to select and connect specific analog channels to your Simulink model.

After you add an Analog Input block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's AD512 I/O board as an example:

- 1 Double-click the Analog Input block.

The Block Parameters: Analog Input dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box. For example, enter

0.001

- 3 In the **Input channels** box, enter a channel vector that selects the analog input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all eight analog input channels on the AD512 board, enter

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

If you want to use the first three analog input channels, enter

[1,2,3]

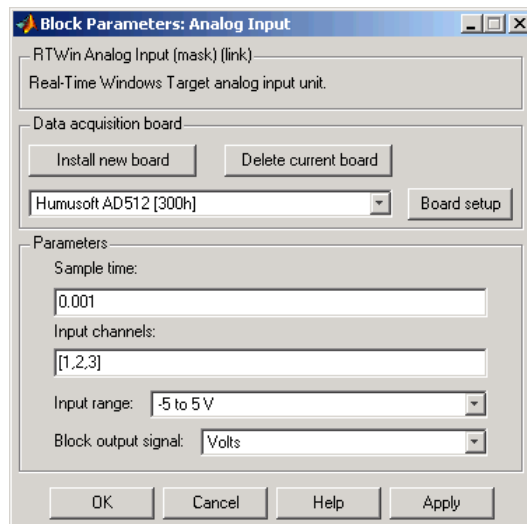
- 4 From the Input range list, choose the input range for all of the analog input channels you entered in the Input channels box. For example, with the AD512 board, choose -5 to 5 V.

If you want the input range to be different for different analog channels, you need to add an I/O block for each different input range.

5 From the Block output signal list, choose from the following options:

- **Volts** — Returns a value equal to the analog voltage.
- **Normalized unipolar** — Returns a full range value of 0 to +1 regardless of the input voltage range. For example, an analog input range of 0 to +5 volts and -5 to +5 volts would both be converted to 0 to +1.
- **Normalized bipolar** — Returns a full range value of -1 to +1 regardless of the input voltage range.
- **Raw** — Returns a value of 0 to $2^n - 1$. For example, a 12-bit A/D converter would return values of 0 to $2^{12} - 1$ (0 to 4095). The advantage of this method is the returned value is always an integer with no roundoff errors.

If you chose Volts, your dialog box will look similar to the next figure.



6 Select one of the following:

- Click the **Apply** button to apply the changes to your model and leave the dialog box open.
- Click the **OK** button to apply the changes to your model and close the Block Parameters: Analog Input dialog box.

Analog Output Block

Real-Time Windows Target I/O blocks allow you to select and connect specific analog channels to your Simulink model.

After you add an Analog Output block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's AD512 I/O board as an example:

- 1 Double-click the Analog Output block.

The Block Parameters: Analog Output dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box. For example, enter

0.001

- 3 In the **Output channels** box, enter a channel vector that selects the analog input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select both analog output channels on the AD512 board, enter

[1,2] or [1:2]

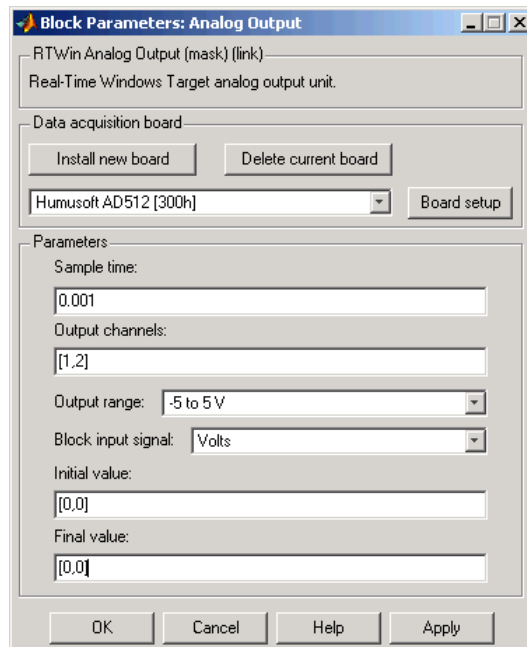
- 4 From the Output range list, choose the input range for all of the analog input channels you entered in the Input channels box. For example, with the AD512 board, choose -5 to 5 V.

If you want the input range to be different for different analog channels, you need to add an I/O block for each different input range.

- 5 From the Block input signal list, choose from the following options:
 - Volts — Expects a value equal to the analog output voltage.
 - Normalized unipolar — Expects a value between 0 and +1 that is converted to the full range of the output voltage regardless of the output voltage range. For example, an analog output range of 0 to +5 volts and -5 to +5 volts would both be converted from values between 0 and +1.

- Normalized bipolar — Expects a value between -1 and +1 that is converted to the full range of the output voltage regardless of the output voltage range.
 - Raw — Expects a value of 0 to $2^n - 1$. For example, a 12-bit A/D converter would expect a value between 0 and $2^{12} - 1$ (0 to 4095). The advantage of this method is the expected value is always an integer with no roundoff errors.
- 6 Enter the initial value for each analog output channel you entered in the **Output channels** box. For example, if you entered [1, 2] in the **Output channels** box, and you want an initial value of 0 volts, enter [0, 0].
 - 7 Enter a final value for each analog channel you entered in the **Output channels** box. For example, if you entered [1, 2] in the **Output channels** box, and you want final values of 0 volts, enter [0, 0].

If you chose Volts, your dialog box will look similar to the next figure.



8 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the Block Parameters: Analog Output dialog box.

Digital Input Block

Real-Time Windows Target I/O blocks allow you to select and connect specific digital lines or digital channels to your Simulink model.

After you have added a Digital Input block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's AD512 I/O board as an example:

1 Double-click the Digital Input block.

The Block Parameters: Digital Input dialog box opens.

2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box. For example, enter

0.001

3 In the **Input channels** box, enter a channel vector that selects the digital input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all eight digital input channels on the AD512 board, enter

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

If you want to use the first four digital input lines, enter

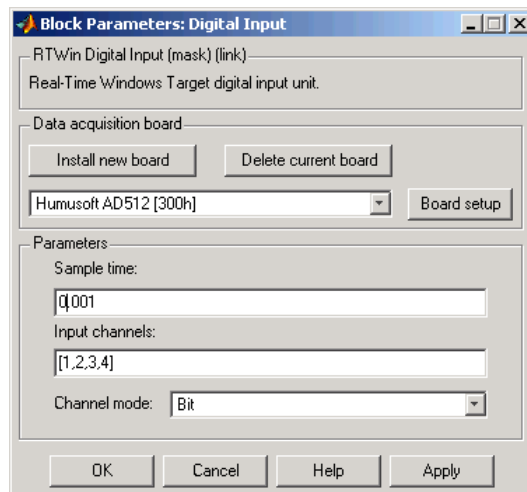
[1,2,3,4]

If you have one 8-bit digital channel, enter [1]. If you have two 8-bit digital channels, enter [1 9], and from the Channel mode list, choose Byte.

4 From the Channel mode list, choose one of the following options:

- **Bit** — Returns a value of 0 or 1.
- **Byte** — Groups eight digital lines into one digital channel and returns a value of 0 to 255.

If you chose **Bit**, your dialog box will look similar to the next figure.



5 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the Block Parameters: Digital Input dialog box.

Digital Output Block

Real-Time Windows Target I/O blocks allow you to select and connect specific digital lines or digital channels to your Simulink model.

After you have added a Digital Output block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's AD512 I/O board as an example:

- 1 Double-click the Digital Output block.

The Block Parameters: Digital Output dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box. For example, enter

0.001

- 3 In the **Output channels** box, enter a channel vector that selects the digital output channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all eight digital output channels on the AD512 board, enter

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

If you want to use the first four digital output lines, enter

[1,2,3,4]

If you have one 8-bit digital channel, enter [1]. If you have two 8-bit digital channels, enter [1 9], and from the Channel mode list, choose Byte.

- 4 From the Channel mode list, choose from one of the following:

- Bit — Expects a value of 0 or 1.
- Byte — Expects a value of 0 to 255 that is converted to one digital channel of eight digital lines.

- 5 Enter the initial values for each digital output line or channel you entered in the **Output channels** box. For example, if you entered [1,2,3,4] in the **Output channels** box, and you want initial values of 0 and 1, enter

[0,0,1,1]

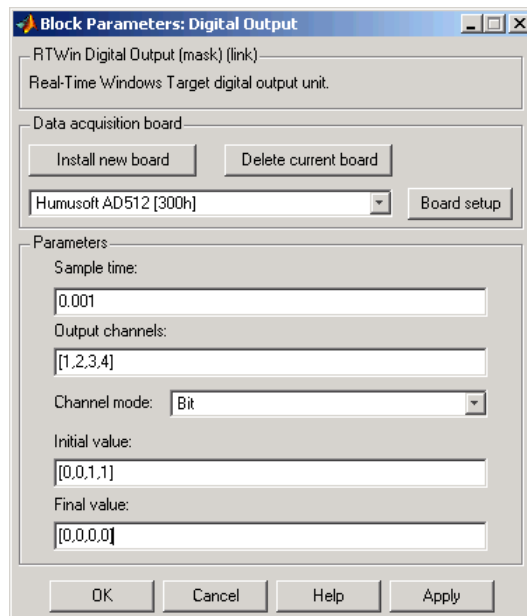
If you choose Byte from the Channel mode list, enter a value between 0 and 255 for each digital output channel. For example, for one byte (8 digital lines) with an initial value of 25, enter [25]. For two bytes (16 digital lines) with initial values of 25 and 50, enter [25 50].

- 6** Enter a final value for each digital output channel you entered in the **Output channels** box. For example, if you entered [1, 2, 3, 4] in the **Output channels** box, and you want final values of 0, enter

[0,0,0,0]

If you choose Byte from the Channel mode list, enter a value between 0 and 255 for each digital output channel.

If you chose Bit, your dialog box will look similar to the next figure.



- 7** Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the Block Parameters: Digital Output dialog box.

Counter Input Block

This Real-Time Windows Target I/O block allows you to select and connect specific counter input channels to your Simulink model.

After you have added a Counter Input block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's MF604 I/O board as an example:

- 1 Double-click the Counter Input block.

The Block Parameters: Counter Input dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box. For example, enter

0.001

- 3 In the **Input channels** box, enter a channel vector that selects the counter input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all four counter input channels on the MF604 board, enter

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

- 4 From **Reset after read**, which determines if the counter should be reset to zero after its value has been read, choose one of the following options:
 - **never** — Do not reset after reading.
 - **always** — Always reset after reading.
 - **level** — Reset after reading if block input is nonzero. This will add an input to the Counter Input block.
 - **rising edge** — Reset after reading if block input changes from zero to nonzero between the last two successive readings. This will add an input to the Counter Input block.

- **falling edge** — Reset after reading if the block input changes from nonzero to zero between last two successive readings. This will add an input to the Counter Input block.
- **either edge** — Reset after reading if the block input changes either from zero to nonzero or from nonzero to zero between the last two successive readings. This will add an input to the Counter Input block.

5 From **Clock input active edge**, which determines which clock edge should increment the counter, select

- **rising** — Low to high transitions
- **falling** — High to low transitions

Not all counter chips support selecting the input edge. In this case, the pull-down menu will reflect the supported option only.

6 From **Gate input functionality**, which defines the action of the counter's gate pin, select

- **none** — Gate is disabled.
- **enable when high** — Counting is disabled when the gate is low and enabled when the gate is high.
- **enable when low** — Counting is disabled when the gate is high and enabled when the gate is low.
- **start on rising edge** — Counting is disabled until low to high transition of the gate occurs.
- **start on falling edge** — Counting is disabled until high to low transition of the gate occurs.
- **reset on rising edge** — Counter is reset when low to high transition of the gate occurs.
- **reset on falling edge** — Counter is reset when high to low transition of the gate occurs.
- **latch on rising edge** — The count of the counter is remembered when low to high transition of the gate occurs.
- **latch on falling edge** — The count of the counter is remembered when high to low transition of the gate occurs.

- latch & reset on rising edge — The count of the counter is remembered and then the counter is reset when low to high transition of the gate occurs.
- latch & reset on falling edge — The count of the counter is remembered and then the counter is reset when high to low transition of the gate occurs.

Not all counter chips support all gate modes. Only supported gate modes are shown in the pull-down menu.

7 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the Block Parameters: Counter Input dialog box.

Encoder Input Block

This Real-Time Windows Target I/O block allows you to select and connect specific encoder input channels to your Simulink model.

After you have added an Encoder Input block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's MF604 I/O board as an example:

1 Double-click the Encoder Input block.

The Block Parameters: Encoder Input dialog box opens.

2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box. For example, enter

0.001

- 3** In the **Input channels** box, enter a channel vector that selects the encoder input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all four encoder input channels on the MF604 board, enter

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

- 4** Encoders typically use two sets of stripes, shifted in phase, to optically detect the amplitude and direction of movement. The **Quadrature mode** parameter specifies which encoder stripe edges should be counted.

- `double` — Counts the rising edges from both stripe sets
- `single` — Counts the rising edges from one stripe set
- `quadruple` — Counts rising and falling edges from both stripe sets

Quadruple mode yields four times more pulses per revolution than the single mode. Therefore, quadruple is more precise and is recommended unless other parameters dictate otherwise.

- 5** The encoder interface chip has a reset pin in addition to encoder inputs. This pin is usually connected to the index output of the encoder. However, it can be connected to any signal or not be used at all. The **Reset input function** specifies the function of this pin.

- `gate` — Enables encoder counting
- `reset` — Level reset of the encoder count
- `rising edge index` — Resets the encoder count on the rising edge
- `falling edge index` — Resets the encoder count on the falling edge

- 6** The encoder interface chip has a built-in lowpass filter that attempts to filter out any high frequencies, which are interpreted as noise. The **Input filter clock frequency** is the cutoff frequency (Hz) of this filter. The cutoff frequency you specify is rounded to the nearest frequency supported by the chip.

If the encoder is moving slowly and high-frequency noise is present, employ the filter to eliminate the noise. This keeps the noise from being counted as encoder pulses. If the encoder is moving quickly, the filter can filter out all of the high-frequency pulses, including those you want to count. In this

case, consider leaving the filter disabled by setting the cutoff frequency to Inf.

7 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the Block Parameters: Encoder Input dialog box.

Other Input and Other Output Blocks

The Real-Time Windows blocks Other Input and Other Output are used for interfacing input and output signals that the six previously described blocks do not accommodate. The Other Input and Other Output blocks are used rarely, and for only a few drivers. See the driver documentation for details.

Output 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.

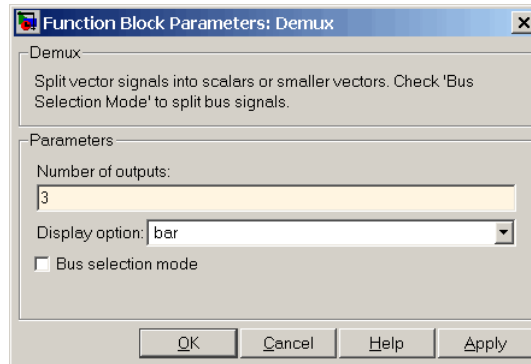
After you add and configure an I/O driver block in your Simulink model, you can separate and connect the output signals from the blocks:

1 In the Simulink window, and from the **View** menu, click **Library Browser**.

The Simulink Library Browser opens.

2 In the **Simulink** library, click **Signal Routing**. From the list in the right column, 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 driver block, enter 3 in the **Number of outputs** box.



4 Click **OK**.

5 Finish making connections and selecting display options.

- Connect the Analog Input block to the Demux block input.
- Connect each of the Demux block output lines to the input of other blocks.
- In the Simulink window, and from the **Format** menu, click **Port/Signal Displays > Wide Nonscalar Lines**, and click **Signal Dimensions**.

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

Variations with Channel Selection

For a better understanding of how to specify device settings when using both analog and digital signals, this section uses the I/O board DAS-1601 from Keithley-Metrabyte 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]

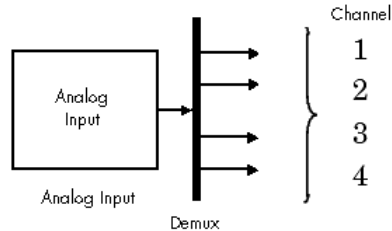
Input vector with single-ended analog — Now, assume your DAS-1601 board is configured to be single-ended analog input. In this case, 16 analog input channels are supported. The complete input vector is

[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]

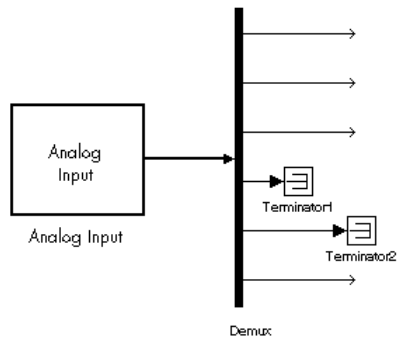
The next figure shows the resulting block diagram.



We do not recommend specifying 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, we recommend specifying only the set of channels that are actually needed for your model.

Using Analog I/O Drivers

- “I/O Driver Characteristics” on page 4-29
- “Normalized Scaling for Analog Inputs” on page 4-30

Control systems have unique requirements for I/O devices that Real-Time Windows Target supports.

For information about writing custom I/O device drivers to work with Real-Time Windows Target, see Appendix A, “Custom I/O Driver Blocks Reference”.

I/O Driver Characteristics

Real-Time Windows Target uses off-the-shelf I/O boards provided by many hardware vendors. These boards are often used for data acquisition independently of Real-Time Windows Target. 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 Real-Time Windows Target.

In data acquisition applications, data is often collected in a burst or frame consisting of many points, perhaps 1,000 or possibly more. The burst of data becomes available once the final point is available. This approach is not suitable for use in automatic control applications since it results in latencies equal to $1000 * T_{\text{sample}}$ for each point of data.

In contrast, drivers used by Real-Time Windows Target capture a single point of data 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 the reason why a board that specifies a maximum sample rate (for data acquisition) might be described as achieving sample rates well in excess of the rates that are achievable in Real-Time Windows Target. For data acquisition, such boards are usually acquiring data in bursts and not in a point-by-point fashion, which is more appropriate for stable control systems.

Normalized Scaling for Analog Inputs

Real-Time Windows Target 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.

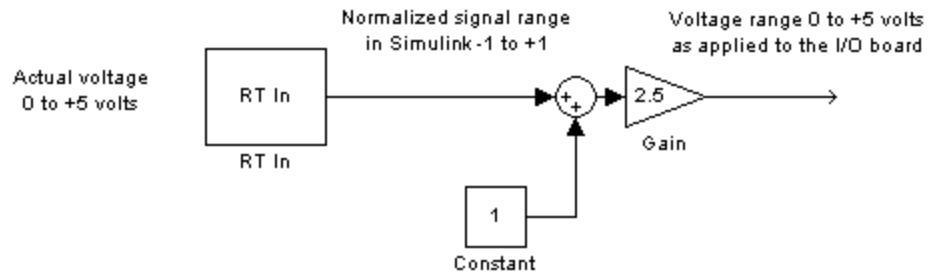
If you prefer unrounded integer values from the analog-to-digital conversion process, you can choose Raw.

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.

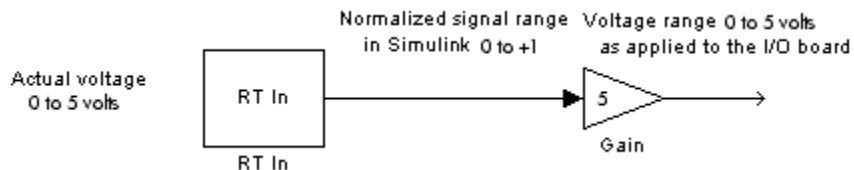


0 to +5 Volts and Normalized Unipolar

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.

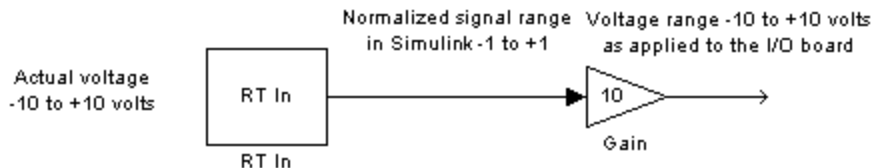


-10 to +10 Volts and Normalized Bipolar

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.

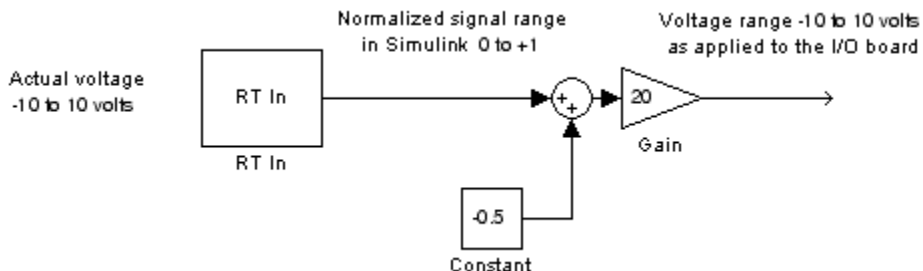


-10 to +10 Volts and Normalized Unipolar

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.

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

In your block diagram, do this as follows.



Normalized 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 in Simulink using a Gain block, Constant block, and Summer block depending on the selected voltage range.

Troubleshooting

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

Building Older Models (p. 5-2)	Building older models
Plots Not Visible in Simulink Scope Block (p. 5-3)	Plots not visible in Scope blocks
Failure to Connect to Target (p. 5-4)	Target connection failure
Sample Time Too Fast (p. 5-5)	Sample times are too small
S-Functions Using Math Functions (p. 5-6)	S-functions that use math functions
Restricted Space for S-Function Local Variables (p. 5-7)	S-function local variable restriction

Building Older Models

If you are building an older model for Real-Time Windows Target, you might get a message like the following:

```
"Real-Time Workshop 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, Real-Time Workshop 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 `rtwinconfigset` command, as follows:

```
rtwinconfigset('<model_name>')
```

Plots Not Visible in Simulink Scope Block

For data to plot correctly in a Simulink Scope block, you must specify the following:

- **External mode** selected from the **Simulation** menu in the Configuration Parameters dialog
- **Connect to target** selected from the **Simulation** menu
- Select one or more signals for capture (designated with "X") in the External Signal & Triggering dialog box from the **Tools > External Mode Control Panel** menu.
- **Duration * Fixed Step Size** close to or less than the X range in the Scope block
- Correct mode (one-shot vs. normal)
- Appropriate signal levels to allow triggering
- Y range on Simulink Scope block axes large enough to span the signal amplitude
- X range
- **Arm when connect to target** in the External Signal & Triggering dialog box or **Arm Trigger** in the **External Mode Control Panel**
- **Start real-time code** selected from the **Simulation** menu

If you are unable to see signals plotted in your Simulink Scope blocks after all of the above items have been selected, your system might have insufficient CPU time. To determine CPU utilization, type `rtwho`. The `rtwho` command returns information about MATLAB performance. The value returned is an indicator of how much loading your model places on the CPU. If Scope blocks fail to plot, this can be an indication that insufficient time is available between sample intervals to allow data to be transferred back to the MATLAB environment where the plotting is performed. To test for this condition, you can run one of the demonstration models, or you can try running your model at a significantly slower rate to determine whether this is the cause. We recommend that MATLAB performance not fall below 80%.

Failure to Connect to Target

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 successfully, we recommend that you delete `.mk` and `.obj` files from the Real-Time Workshop project directory, and then select **Build** from the **Tools** menu.

Possible Problem — When trying to connect to the target, the Simulink Diagnostic dialog box displays

```
External mode MEX-file "win_tgt" does not exist or is not on the  
MATLAB path.
```

Solution — Real-Time Windows Target Versions 1.0 and 1.5 used the MEX-file `win_tgt`. For Real-Time Windows Target Version 2.2 and later, the MEX-file name was changed to `rtwinext`. If you create a new Simulink model, the new filename is entered correctly. If you have Simulink models where you used Real-Time Windows Target 1.0 or 1.5, you need to change the filename using the following procedure:

- 1** In the Simulink window, and from the **Tools** menu, click **External Mode Control Panel**.
- 2** On the External Mode Control Panel dialog box, click the **Target interface** button.
- 3** In the **MEX-file for external mode** text box, enter

```
rtwinext
```

- 4** Click **OK**.

Sample Time Too Fast

During a run, you might not see any output in the Scope window. This could indicate that the sample time is too small. In the MATLAB Command Window, type

```
rtwho
```

Check the value for MATLAB performance. A value less than 80% indicates that your sample time might be too small.

In general, we recommend that you start by choosing a slow sample rate. For example, select a sample time of 0.01 second, and confirm that your system runs correctly and plots are displayed. Should you select a sample rate that exceeds the capability of your computer, an error message is displayed and real-time execution is terminated. If this occurs, select a slower sample rate. Then rebuild the model, connect to the target, and start the real-time application again. You must rebuild the real-time application after changing the sample time.

Check the MATLAB performance value returned when you type `rtwho`. If MATLAB performance is in the range of 98% or so, then consider decreasing your sample time by one order of magnitude.

If you notice either slow updates of Scope blocks or a complete failure to plot data in the Scope blocks, you might be reaching the upper threshold for the sample rate on your hardware. Plotting data has a lower priority than execution of your real-time application.

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 Real-Time Windows Target header to your S-function. For example, add

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

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

Restricted Space for S-Function Local Variables

The Real-Time Windows kernel provides stack space for at most 2 KB of local variables. Code generated by Real-Time Windows Target never exceeds this limit, but a user-written S-function could do so. Executing such code can cause the application to fail or the computer to reboot. If you create your own S-functions, ensure that local variables do not exceed the stack size limit of 2 KB.

Custom I/O Driver Blocks Reference

Custom I/O device drivers can be used in combination with Real-Time Windows Target. We do not recommend using Analog Input, Analog Output, Digital Input, or Digital Output drivers as a starting point for creating custom device drivers. You can write custom I/O device drivers to work with Real-Time Windows Target.

I/O Register Access from S-Functions Limitation (p. A-2)	Illustrates sample code for I/O register access from S-functions limitation
Incompatibility with Win32 API Calls (p. A-3)	Describes incompatibility with Win32 API calls
Unsupported C Functions (p. A-4)	Lists unsupported C functions
Supported C Functions (p. A-5)	Lists supported C functions

I/O Register Access from S-Functions Limitation

For Windows 2000 and Windows XP, drivers can access I/O registers only from the real-time kernel and not from Simulink. To ensure that drivers do not attempt to access I/O registers from Simulink S-functions, enter code fragments like the following as appropriate:

```
#ifndef MATLAB_MEX_FILE
/* we are in RTWin kernel, safe to do board I/O */
#else
/* we are in Simulink, don't do board I/O */
#endif
```

Incompatibility with Win32 API Calls

The Real-Time Windows Target 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 Windows operating system to allow Windows applications or any software using the Win32 API to run.

As a result, **software that uses the Win32 API cannot be executed as a component of your real-time application.** Any software you use to write I/O drivers must not have any calls to the Win32 API.

Unsupported C Functions

If you create your own custom I/O driver blocks, you should first check for C functions that are supported by Real-Time Windows Target.

Functions that use the Windows operating system are not supported with Real-Time Windows Target. This is because the kernel intercepts the system clock and first runs the real-time application. If there is time left before the next sample time, the kernel might allow a Windows application or function to run.

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** — printf, fprintf, sprintf, vsprintf, vprintf, vsprintf, fscanf, scanf, sscanf
- **Process management** — spawn, exit, abort, atexit
- **Signals and exceptions** — signal, longjmp, raise
- **Time functions** — clock, time, difftime, asctime, ctime, difftime, gmtime, localtime, mktime, strftime
- **Win32 API functions** — *No Windows API functions are supported.*

Supported C Functions

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

- **Data conversion** — `abs`, `atof`, `atoi`, `atol`, `itoa`, `labs`, `ltoa`, `strtod`, `strtol`, `strtoul`, `ultoa`
- **Memory allocation** — `calloc`, `free`, `malloc`

Note Memory allocation is not an operation that can be done in real time. To work with Real-Time Windows Target, 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`
- **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`, `isctrl`, `isdigit`, `isgraph`, `islower`, `isprint`, `ispunct`, `isspace`, `isupper`, `isxdigit`, `isxupper`, `isxlower`, `_toascii`, `tolower`, `toupper`
- **Searching and sorting** — `bsearch`, `qsort`
- **Dummy functions** — `exit`, `fprintf`, `printf`

Examples

Use this list to find examples in the documentation.

Simulink Model Examples

“Entering Configuration Parameters for Simulink” on page 3-6

“Entering Scope Parameters for Signal Tracing” on page 3-7

“Running a Non-Real-Time Simulation” on page 3-10

“Specifying a Default Real-Time Windows Target Configuration Set” on page 3-11

Real-Time Application Examples

“Entering Simulation Parameters for Real-Time Workshop” on page 3-13

“Entering Scope Parameters for Signal Tracing” on page 3-16

“Creating a Real-Time Application” on page 3-18

“Entering Additional Scope Parameters for Signal Tracing” on page 3-19

“Running a Real-Time Application” on page 3-21

“Running a Real-Time Application from the MATLAB Command Line” on page 3-24

Signal Logging to MATLAB Workspace Examples

“Entering Scope Parameters” on page 3-26

“Entering Signal and Triggering Properties” on page 3-28

“Plotting Logged Signal Data” on page 3-31

Signal Logging to Disk Drive Examples

“Entering Scope Parameters” on page 3-33

“Entering Signal and Triggering Properties” on page 3-36

“Entering Data Archiving Parameters” on page 3-38

“Plotting Logged Signal Data” on page 3-40

Parameter Tuning Examples

“Changing Model Parameters” on page 3-44

I/O Board Examples

“Installing and Configuring I/O Boards and Drivers” on page 4-2

“Real-Time Windows Target Library” on page 4-9

“Analog Input Block” on page 4-12

“Analog Output Block” on page 4-14

“Digital Input Block” on page 4-16

“Digital Output Block” on page 4-17

“Counter Input Block” on page 4-20

“Encoder Input Block” on page 4-22

“Output Signals from an I/O Block” on page 4-24

A

- A/D. *See* analog-to-digital
- adding
 - Analog Input block 4-9
 - I/O driver blocks 4-9
 - input blocks 4-9
- analog input
 - normalized scaling 4-30
- Analog Input block
 - configuring 4-12
- Analog Output block
 - configuring 4-14
- analog-to-digital
 - channel selection 4-25
- application
 - Real-Time Windows Target 1-6

C

- capturing and displaying signals 1-2
- changing parameters
 - parameter tuning 1-7
- channel selection
 - entering configurations 4-25
- compact PCI
 - installing 4-8
- compatibility
 - with the MathWorks software 1-4
- compiler
 - Open Watcom C/C++ only 1-5
- computer
 - PC-compatible 1-9
- configuring
 - Analog Input block 4-12
 - Analog Output block 4-14
 - Counter Input block 4-20
 - Digital Input block 4-16
 - Digital Output block 4-17
 - Encoder Input block 4-22
 - I/O boards and drivers 4-2

- connecting
 - real-time application 3-19
 - Simulink model 3-19
- Counter Input block
 - configuring 4-20
- creating
 - real-time application 3-18
 - Simulink model 1-12
- custom I/O drivers
 - incompatible with Win32 A-3

D

- D/A. *See* digital-to-analog
- data archiving parameters
 - entering 3-38
 - logging data to disk drive 3-38
- data buffers 1-14
- default configuration set 3-11
- demo library
 - opening 2-18
 - Real-Time Windows Target 2-18
- Demux block
 - separating I/O signals 4-24
- description
 - Simulink external mode 1-13
- device drivers
 - channel selection 4-25
 - custom I/O A-1
 - writing custom A-1
- Digital Input block
 - configuring 4-16
- Digital Output block
 - configuring 4-17
- digital-to-analog
 - channel selection 4-25
- directories
 - installed 2-7
 - MATLAB working 2-7
 - project 2-7

- Real-Time Workshop working 2-7
- working 2-7

- disk drive

- plotting logged data 3-40

- signal logging 3-33

E

- Encoder Input block

- configuring 4-22

- entering

- data archiving parameters 3-38

- scope properties 3-33

- signal and triggering properties 3-36

- simulation parameters for Real-Time

- Workshop 3-13

- simulation parameters for Simulink 3-6

- execution

- real-time 1-11

- running in real time 1-12

- external interface MEX-file

- rtwinext 5-4

- win_tgt 5-4

- external mode

- description 1-13

- parameter tuning 1-7

F

- failure to connect

- troubleshooting 5-4

- features

- signal logging 1-6

- signal tracing 1-6

- files

- application 2-7

- external mode interface 2-7

- I/O drivers 2-7

- installed 2-7

- kernel install command 2-7

- make command 2-7

- makefile 2-7

- project directory 2-7

- Real-Time Windows Target directory 2-7

- system target 2-7

- system target file 2-7

- template makefile 2-7

- working directory 2-7

H

- hardware

- system requirements 2-5

I

- I/O blocks

- Analog Input block 4-12

- Analog Output block 4-14

- Counter Input block 4-20

- Digital Input block 4-16

- Digital Output block 4-17

- Encoder Input block 4-22

- input and output 4-9

- separating signals 4-24

- I/O boards

- compact PCI boards 4-8

- configuring 4-2

- installing 4-2

- ISA bus 4-6

- overview 4-2

- PC/104 bus 4-8

- PCI bus 4-7

- PCMCIA 4-8

- I/O drivers

- characteristics 4-29

- configuring 4-2

- installing 4-2

- using 4-29

- input blocks

- adding Analog Input blocks 4-9
 - overview 4-9
- input/output
 - support 1-9
- installing
 - compact PCI 4-8
 - I/O boards 4-2
 - I/O boards and drivers 4-2
 - kernel overview 2-10
 - Real-Time Windows Target 2-7
 - testing installation 2-14
- ISA bus
 - installing 4-6

K

- kernel
 - communication with hardware 1-4
 - installing 2-10
 - scheduler 1-4
 - timer interrupt 1-4
 - uninstalling 2-11

L

- logging
 - data to disk drive 3-33
 - data archiving parameters 3-38
 - data to workspace 3-26

M

- makefile 2-7
- MathWorks
 - compatible software 1-4
 - Simulink blocks 1-4
- MATLAB workspace
 - signal logging 3-26
- memory management
 - limitation on A-5
- model parameters

- changing 3-44
- parameter tuning 3-44

N

- nonreal time
 - simulation 3-10
- normalized scaling
 - analog input 4-30

O

- opening demo library 2-18
- output blocks
 - overview 4-9
- overview
 - I/O boards 4-2
 - input blocks 4-9
 - installing kernel 2-10
 - output blocks 4-9
 - parameter tuning 3-43
 - real-time application 3-13
 - Real-Time Windows Target 1-2
 - system concepts 1-13
 - system requirements 2-5
 - testing installation 2-14

P

- parameter tuning
 - changing model parameters 3-44
 - changing parameters 1-7
 - external mode 1-7
 - feature 1-7
 - overview 3-43
- PCI bus
 - installing 4-7
- PCMCIA bus
 - installing 4-8
- plots not visible
 - troubleshooting 5-3

plotting

- logged data from disk 3-40
- logged data from workspace 3-31

R

real-time

- control 1-2
- execution 1-11
- hardware-in-the-loop 1-2
- signal processing 1-2

real-time application

- and the development process 1-12
- connecting to Simulink model 3-19
- creating 3-18
- overview 3-13
- Real-Time Windows Target 1-6
- Real-Time Workshop parameters 3-13
- scope properties for signal tracing 3-16
- simulation parameters for Real-Time Workshop 3-13
- software environment 1-6
- starting 3-21
- stopping 3-21

real-time kernel

- Real-Time Windows Target 1-4
- scheduler 1-4
- software environment 1-4
- timer interrupt 1-4

Real-Time Windows Target

- application 1-6
- custom I/O device drivers A-1
- demo library 2-18
- development process 1-12
- files 2-7
- installing kernel 2-10
- overview 1-2
- real-time application 1-6
- real-time kernel 1-4
- software environment 1-11

uninstalling kernel 2-11

what is it? 1-2

Real-Time Workshop

- entering simulation parameters 3-13

requirements

- hardware 2-5
- software 2-6

rtvdp.mdl

- Simulink model 2-14

RTWin configuration set 3-11

rtwinext

- external interface MEX-file 5-4

running

- execution in real time 1-12
- real-time application 3-21
- simulation in nonreal time 3-10

S

S-functions

- C-code supported in 2-2
- limitation on I/O register access A-2
- limitation on memory management A-5
- M-code not supported in 2-2
- restricted stack space for local variables 5-7
- using math functions in 5-6
- Win32 calls not usable in 1-4

sample rates

- excessive 2-17

sample time

- too fast 5-5

scope properties

- entering 3-33
- entering for signal tracing 3-16
- for signal logging to disk drive 3-33
- for signal logging to workspace 3-26

separating

- I/O signals 4-24

setting

- initial working directory 2-9

- working directory
 - from MATLAB 2-9
 - from the Desktop icon 2-9
 - signal and triggering
 - entering properties 3-28
 - properties 3-36
 - signal archiving. *See* signal logging
 - signal data
 - plotting from disk drive 3-40
 - plotting from workspace 3-31
 - signal logging
 - entering scope properties 3-26
 - feature 1-6
 - plotting data 3-31
 - signal and triggering properties 3-36
 - to disk drive 3-33
 - to MATLAB workspace 3-26
 - signal logging to disk drive
 - data archiving parameters 3-38
 - signal and triggering properties 3-36
 - signal logging to workspace
 - scope properties 3-26
 - signal and triggering properties 3-28
 - signal tracing
 - feature 1-6
 - scope properties 3-16
 - signals
 - capturing and displaying 1-2
 - simulation
 - nonreal time 3-10
 - running in nonreal time 1-12
 - simulation parameters
 - entering 3-6
 - for Real-Time Workshop 3-13
 - Simulink
 - compatibility 2-2
 - compatible software 1-4
 - required product 2-2
 - running a simulation 3-10
 - Simulink external mode
 - description 1-13
 - parameter tuning 1-7
 - Simulink model
 - and the development process 1-12
 - connect to real-time application 3-19
 - creating 1-12
 - rtvdp.mdl 2-14
 - software
 - system requirements 2-6
 - software environment
 - overview 1-11
 - real-time application 1-6
 - real-time kernel 1-4
 - requirements 2-6
 - starting
 - real-time application 3-21
 - stopping
 - real-time application 3-21
 - support
 - input/output 1-9
 - system concepts
 - data buffers 1-14
 - overview 1-13
 - transferring data 1-14
 - system requirements
 - hardware 2-5
 - overview 2-5
 - software 2-6
 - software environment 2-6
 - system target file 2-7
- T**
- template makefile 2-7
 - testing installation
 - overview 2-14
 - transferring data 1-14
 - troubleshooting
 - failure to connect 5-4
 - incorrect MEX-file 5-4

plots not visible 5-3
sample time too fast 5-5

U

uninstalling
 kernel 2-11
using
 I/O device drivers 4-29

W

Win32

calls not usable in S-functions 1-4
incompatible with I/O drivers A-3
Windows Target. *See* Real-Time Windows Target
working directory
 initial 2-9
 setting
 from MATLAB 2-9
 from the Desktop icon 2-9
 setting initial 2-9
writing customized device drivers A-1