Real-Time Windows Target

For Use with Real-Time Workshop®

Computation

Visualization

Programming

User's Guide

Version 1

How to Contact The MathWorks:

508-647-7000	Phone

508-647-7001	Fax

The MathWorks, Inc. 24 Prime Park Way Natick, MA 01760-1500	Mail

`http://www.mathworks.com`	Web
`ftp.mathworks.com`	Anonymous FTP server
`comp.soft-sys.matlab`	Newsgroup

`support@mathworks.com`	Technical support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`subscribe@mathworks.com`	Subscribing user registration
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

Real-Time Windows Target User's Guide

Printing History: January 1999 First printing New for Version 1.0 (Release 11)

Contents

Foreword

1 Real-Time Windows Target Fundamentals

2 Device Driver Settings

3 Parameter Tuning and External Mode

4 Real-Time Windows Target Troubleshooting

5 Real-Time Windows Target Block Reference

6 Real-Time Windows Target Function Reference

Foreword

Related Products and Documentation

Requirements

The Real-Time Windows Target runs on Microsoft Windows 95, Windows 98, and Windows NT systems.

The Real-Time Windows Target requires:

- MATLAB® 5.3 (Release 11)
- Simulink® 3.0 (Release 11)
- Real-Time Workshop® 3.0 (Release 11)
- Watcom C 11.0 language compiler

See the MATLAB *Application Program Interface Guide* for information on how to ensure proper installation for integration into the MATLAB environment.

What Is MATLAB?

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for *matrix laboratory*. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK

and EISPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called *toolboxes*. Very important to most users of MATLAB, toolboxes allow you to *learn* and *apply* specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

See the MATLAB documentation set for more information.

What Is Simulink?

Simulink is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate, i.e., have different parts that are sampled or updated at different rates.

For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. With this interface, you can draw the models just as you would with pencil and paper (or as most textbooks depict them). This is a far cry from previous simulation packages that require you to formulate differential equations and difference equations in a language or program. Simulink includes a comprehensive block library of sinks, sources, linear and nonlinear components, and connectors. You can also customize and create your own blocks.

Models are hierarchical, so you can build models using both top-down and bottom-up approaches. You can view the system at a high-level, then double-click on blocks to go down through the levels to see increasing levels of model detail. This approach provides insight into how a model is organized and how its parts interact.

After you define a model, you can simulate it, using a choice of integration methods, either from the Simulink menus or by entering commands in MATLAB's command window. The menus are particularly convenient for interactive work, while the command-line approach is very useful for running a batch of simulations (for example, if you are doing Monte Carlo simulations or want to sweep a parameter across a range of values). Using scopes and other display blocks, you can see the simulation results while the simulation is running. In addition, you can change parameters and immediately see what happens, for "what if" exploration. The simulation results can be put in the MATLAB workspace for postprocessing and visualization.

Model analysis tools include linearization and trimming tools, which can be accessed from the MATLAB command line, plus the many tools in MATLAB and its application toolboxes. And because MATLAB and Simulink are integrated, you can simulate, analyze, and revise your models in either environment at any point.

The *Using Simulink* document describes how to work with Simulink. It explains how to manipulate Simulink blocks, access block parameters, and connect blocks to build models. It also provides reference descriptions of each block in the standard Simulink libraries.

What Is the Real-Time Workshop?

The Real-Time Workshop (RTW) is an automatic C language code generator for Simulink. It produces C code directly from Simulink block diagram models and automatically builds programs that can be run in real time in a variety of environments, including real-time hardware such as microcontrollers or digital signal processors. You can run the Real-Time Workshop:

- Bareboard (meaning no real-time operating system present) on real-time hardware
- Interfaced with a real-time operating system

How to Use This Guide

Typographical Conventions

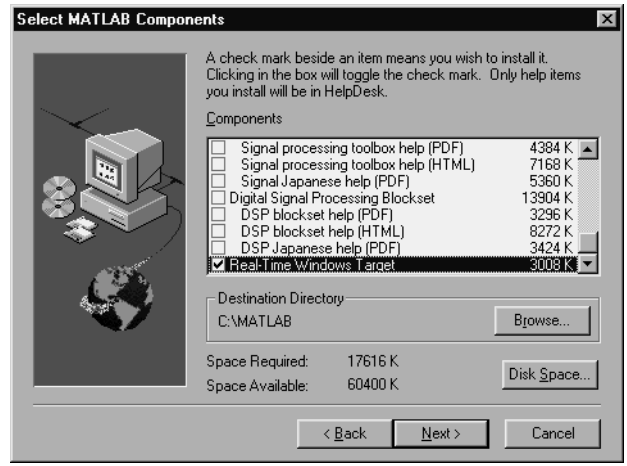
To Indicate...	This Guide Uses...	Example
New terms	*Italics*	An *array* is an ordered collection of information.
Keys, menu names, menu names, and GUI controls	**Boldface** with an initial capital letter	Choose the **File** menu.
Function names, variables, example code, user-entered text	`Monospace` type	The `sfopen` command opens a model.

Installation

Your MATLAB *Installation Guide for PC* provides essentially all of the information you need to install the Real-Time Windows Target.

Prior to installing the Real-Time Windows Target, you must obtain a License File or Personal License Password from The MathWorks. The License File or Personal License Password identifies the products you are permitted to install and use.

As the installation process proceeds, a screen similar to this, where you can indicate which products to install, is displayed.



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

Licensed Product	Prerequisite Products	Additional Information
Simulink 3.0	MATLAB 5.3	Allows installation of Simulink
Real-Time Workshop	Simulink 3.0	Allows installation of the Real-Time Windows Target
Real-Time Windows Target	Real-Time Workshop 3.0	Requires a Watcom C 11.0 compiler

If you experience installation difficulties and have Web access, connect to the MathWorks home page (`http://www.mathworks.com`). Look for the license manager and installation information under the `Tech Notes/FAQ` link under `Tech Support Info`.

Installing and Removing the Real-Time Kernel

A key component of the Real-Time Windows Target is a real-time kernel that interfaces with the Microsoft Windows operating system. The kernel enables the Real-Time Windows Target to assign the highest priority of execution to your real-time executable, which is created by the Real-Time Workshop.

After you have installed the kernel, it remains idle, which allows Windows to control the execution of any standard Windows application (internet browsers, word processors, MATLAB, etc.). 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 may need servicing.

The MATLAB installer automatically sets up the kernel. To complete the process, you must restart your computer if you receive a message telling you to do so. Once the kernel is installed, you can leave it installed.

You can manually install or remove the kernel by running the M-file `rtwintgt.m`. To install the kernel manually, type

```
rtwintgt -install
```

at the MATLAB prompt. This will initiate the kernel. See “I/O Board Support” on page 1-13 for a list of supported boards.

To check that the Real-Time Windows Target has correctly installed the kernel, type

```
rtwho
```

at the MATLAB prompt. You will see something like the following (depending on what board you’ve installed).

```
Real Time Windows Target version 1.00 (C) The MathWorks, Inc.
1998
MATLAB performance = 100.0%
Kernel timeslice period = 1 ms
DRIVERS:                          Name    Address   Inputs  Outputs

                               HUMUSOFT AD512      0x300     16      10
```

If you encounter any problems with the Real-Time Windows Target, you can remove the board's kernel by typing

```
rtwintgt -uninstall
```

at the MATLAB prompt.

Note Any time you install or remove the kernel from your system, you must reboot your computer for the action to be recognized.

Installation Issues

These are the most common problems that arise during installation:

- You do not have administrator privileges on NT — Administrator privileges are required for this operation. Please contact your system administrator.
- You have a multiprocessor machine and run NT — The current version of Real-Time Windows Target cannot run on multiprocessor machines.

1

Real-Time Windows Target Fundamentals

Introduction

The Real-Time Windows Target is software that allows you to run C code generated by Real-Time Workshop on a PC in real time. In this environment, your PC is the host for MATLAB, Simulink, and the Real-Time Workshop. Once C code is generated and compiled, your same PC, running Windows 95, Windows 98, or Windows NT, is then used as a target for running the generated code. Typical applications for the Real-Time Windows Target include real-time control, signal processing, and hardware-in-the-loop simulation.

The generated code runs fast (in real time, at ring zero) under Windows 95, Windows 98, or Windows NT operating systems using standard yet cost-effective I/O boards. When running your models in real time, Real-Time Windows Target captures a sample of 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 via an output channel on your I/O board. The Real-Time Windows Target uses the unique real-time capabilities of a special kernel for each of the supported Windows operating systems, combined with the speed of compiled C code. It provides a custom Simulink block library and more than 60 ready-to-use hardware I/O drivers. Signals may be captured and graphed in Simulink Scope blocks by using Simulink's external mode enabling you to observe the behavior of your real-time system. Simulink's external mode also allows you to alter parameters by simply editing the block diagram while running Simulink in external mode. New parameter values are automatically transferred to the compiled version of your block diagram during real-time execution.

Note The Real-Time Windows Target requires the Watcom 11.0 C/C++ compiler.

The Real-Time Windows Target includes a set of source files, binaries for I/O device driver libraries, a template makefile, and a MEX-file interface. After you create your block diagram model in Simulink, you use the Real-Time Workshop to create a real-time model by clicking the **Build** button on the **Real-Time Workshop** page of the **Simulink Parameters** dialog box. The MEX-file interface module is used to allow Simulink's external mode to export 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.

This guide assumes that you are familiar with the basics of Simulink and the Real-Time Workshop. Refer to *Using Simulink*, *Writing S-Functions*, and the *Real-Time Workshop User's Guide* for information about these products. For information about MEX-files, refer to the *Application Program Interface Guide*.

Files Included with the Real-Time Windows Target

The files included with the Real-Time Windows Target are located in your *matlabroot*\TOOLBOX\RTW\WINDOWS and in the *matlabroot*\RTW\C\WINDOWS directories. This table lists the files and provides a description of each.

Table 1-1: Contents of the Real-Time Windows Target

Filename	Description
win_watc.tlc	System target file that specifies the code generation format used by the Real-Time Workshop
win_watc.tmf	Template makefile for automatic makefile generation by the Real-Time Workshop
win_tgt.dll	MEX-file for communicating between Simulink's external mode and the Real-Time Windows Target kernel
*.c, *.h	Source and include files in C language
*.obj	Object files with glue routines

Getting Started

A Watcom 11.0 C/C++ compiler is required (not included) for using Real-Time Windows Target. During installation of your Watcom 11.0 C/C++ compiler, please be sure to specify DOS target in addition to Windows target to have the necessary libraries available for linking.

Confirming Installation of the Real-Time Windows Target Kernel

The Real-Time Windows Target includes a set of M-file commands for accessing the kernel and displaying information. To confirm that the kernel for Real-Time Windows Target is correctly installed on your machine, at the MATLAB command line type

```
rtwho
```

MATLAB should return the following message:

```
Real Time Windows Target version 1.00 (C) The MathWorks, Inc.
1998
MATLAB performance = 100.0%
Kernel timeslice period = 1 ms
```

This message is telling you that MATLAB and other nonreal-time application software (e.g., a word processor) are able to run at 100% performance because no real-time models are currently executing on your PC. When a real-time model is executing, you will observe that messages for MATLAB performance will show values below 100%. This is an indication of two things:

- First, it indicates that MATLAB is able to communicate with the kernel, which suggests that the kernel is correctly installed.
- Second, it indicates that the real-time model is running at a higher priority than any other application on your PC.

These two facts mean that your real-time model is running as the highest priority process and, accordingly, it will be serviced before any other process to ensure that the real-time model runs deterministically (e.g., the specified sample interval is achieved). Model execution may be set to run at, for example, 0.001 seconds (i.e., a 1 millisecond sample rate). The actual time it takes the CPU to compute one sample step for the model code may take less than 100 microseconds, leaving 900 microseconds or more CPU time available before the

next model step is required. The topic of sample intervals and available CPU time will be discussed in more depth in later sections. Note that small models can run at sample rates well beyond 1 millisecond sample intervals. Depending on your particular hardware, maximum sample rates for a minimum model are likely to exceed 20 KHz.

Testing for Correct Installation of Your Watcom 11.0 C/C++ Compiler

As an initial test to determine whether the Watcom compiler is correctly installed, a demo model is provided with Real-Time Workshop settings preset and saved in the model. Start by typing `rtvdp` at the MATLAB command line. This opens a demonstration example that does not require any I/O boards. The model runs the same dynamics as the Simulink `vdp` model, with one subtle but very important difference — execution is synchronized to a real-time clock. Once you have this model running, you can easily confirm that you are running in real time by comparing the elapsed time to run, say, a 20 second simulation versus the elapsed time on your wrist watch. In contrast, the Simulink `vdp` model normally runs faster than real time and its execution is not synchronized to a real-time clock.

Having opened the model `rtvdp`, click **RTW Build** from the **Tools** menu. Alternatively, you could click **Build** on the **Real-Time Workshop** page of the **Simulation Parameters** dialog box. This directs the Real-Time Workshop to generate C code for the model `rtvdp`. Real-Time Workshop will then attempt to build a real-time executable to run on the PC target. This provides a simple test to confirm that your Watcom 11.0 C/C++ compiler is correctly installed for use with Real-Time Windows Target. Figure 1-1 below shows the model `rtvdp` with the **RTW Build** option selected.

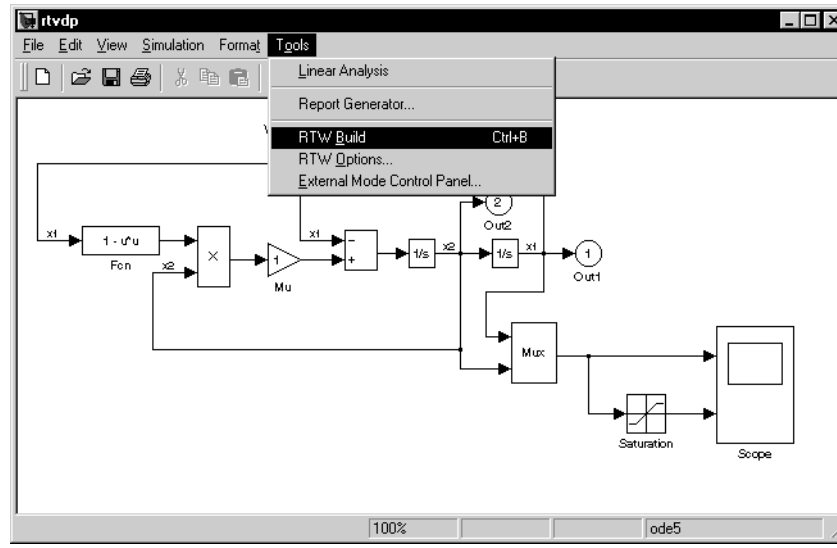


Figure 1-1: Testing Installation with the rtvdp Model

You will observe a series of messages displayed at the MATLAB command line starting with

```
### Starting RTW build procedure for model: rtvdp
### Invoking Target Language Compiler on rtvdp.rtw
```

If your model is able to build successfully, the message string will end with

```
Compiling rtvdp.c
Creating linker response file rtvdp.lnk
Linking model rtvdp
Model rtvdp.rtd succesfully created
### Successful completion of RTW build procedure for model: rtvdp
```

If you observe an error message, this is a likely indication that the Watcom 11.0 C/C++ compiler has not been installed with the components needed by Real-Time Windows Target. Please reinstall the Watcom 11.0 C/C++ compiler. If the model `rtvdp` does not build successfully when using preset **Real-Time Workshop** page settings for Real-Time Windows Target, it is necessary to resolve this problem before going further. The most likely cause is incorrect

installation of either MATLAB, Simulink, the Real-Time Workshop, or the Real-Time Windows Target, or of the Watcom 11.0 C/C++ compiler.

Starting Real-Time Execution

Having successfully built the real-time model for `rtvdp`, you can immediately test this model by running it in real time and monitoring signals during real-time execution. To start execution of the `rtvdp` model, go to the **Simulation** pull-down menu and be sure **External** is selected. Now select **Connect to target**. You will observe that the Simulation menu options now allow you to **Start real-time code** or **Disconnect from target**. Figure 1-2 below shows the `rtvdp` model with **Connect to target** selected.

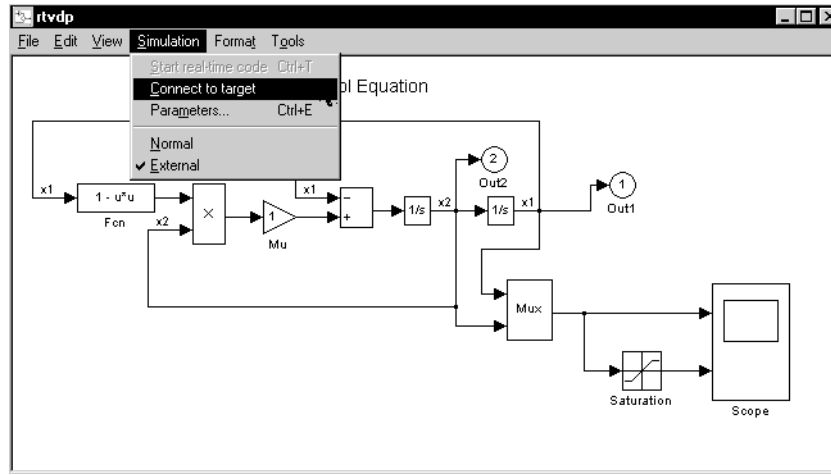


Figure 1-2: Using Simulink External Mode, Connect to Target

Select **Start real-time code**. After a brief delay, usually fewer than four seconds, you should see plots starting to appear in the plot window of the Simulink Scope. This delay is not in execution, which runs in real time, but in the display on the scope. While running this model you should notice that plots are synchronized to real time. You can observe this by comparing against elapsed time on your wrist watch. After trying this, repeat the test with any Simulink model by running a simulation directly in Simulink, without using the Real-Time Windows Target.

Simulink simulations run asynchronously with respect to real time. Depending on your model size and complexity, and the specified sample rate, a Simulink simulation may run significantly faster or slower than real time.

Figure 1-3 below shows the `rtvdp` model with **Start real-time code** selected from the **Simulation** menu.

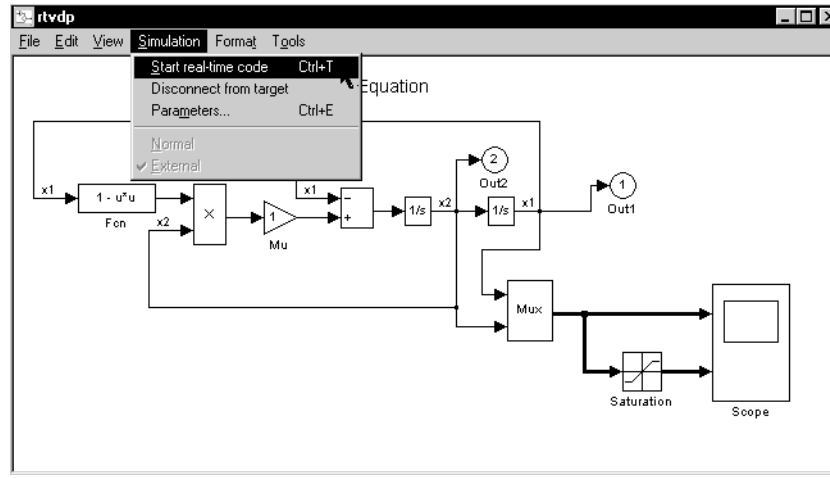


Figure 1-3: Start Real-Time Simulation for the Model rtvdp

With the model `rtvdp` running (while using all default values including step size), once again type `rtwho`. You will observe a message similar to the following:

```
Real Time Windows Target version 1.00 (C) The MathWorks, Inc. 1998
MATLAB performance = 98.5%
Kernel timeslice period = 0.999 ms
TIMERS:  Number      Type     Period   Running

            1          In      0.01      Yes

HISTORIES:      Name         Size      Mode      Link

        _history_ext        1 by 1       0

DRIVERS:                              Name    Address   Inputs  Outputs

                          Real Time Model          0      1       0
```

This message says that the real-time model (which runs at the highest priority, i.e., ring zero) is able to complete the execution of the `rtvdp` model within the 0.01 second sample interval (100 Hz). This is a slow sample rate for the Real-Time Windows Target. You will also notice that the MATLAB performance = 98.5% tells us that MATLAB and other Windows applications are able to get nearly 100% of the CPU when you attempt to use them. We highly recommend that you try never to run with less than a minimum value of MATLAB performance of 50%. Below this value, it is possible that the switch time between processes has such a load on the CPU that you are not able to exchange messages with the real-time model and performance will not be acceptable.

Note In addition to `rtvdp`, there are other demos available. Typing `rtwtdemo` at the MATLAB prompt brings up a GUI with the available demos.

Stopping Real-Time Execution

After starting a real-time simulation the Simulation menu options allow you to select **Stop real-time code** or **Disconnect from target**. Since the Real-Time Windows Target is a unique target in that your host is also your target computer, the two commands are equivalent. You can choose either command and the effect will be the same; execution of your real-time model will stop.

Real-Time Windows Target Block Library

Real-Time Windows Target includes a small block library consisting of a source block (RT In), a sink block (RT Out), an Adapter block, and a collection of four demonstration examples. Typing `rtwinlib` at the MATLAB prompt brings up this window.

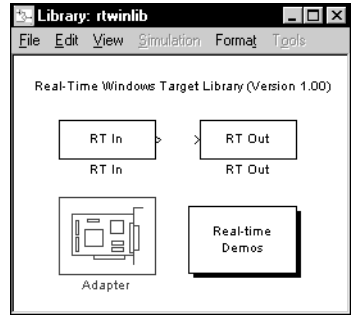


Figure 1-4: Real-Time Windows Target Block Library rtwinlib

The Adapter block is a special component that is not physically wired into other blocks in your block diagram. The purpose of the Adapter block is to provide a reference to a particular I/O board with a valid base address. Multiple Adapter blocks can be placed in a block diagram. Each instance of an Adapter block must have a unique name: Adapter, Adapter1, Adapter2, and so on. Clicking on an Adapter block brings up this dialog box.

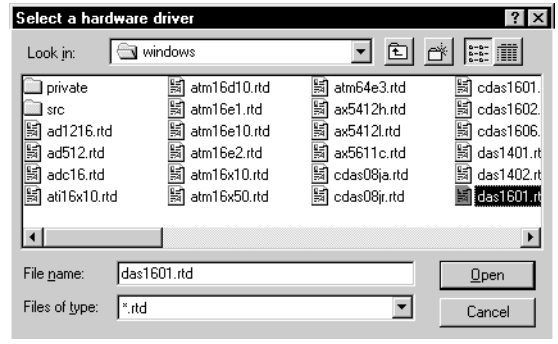


Figure 1-5: Selecting an I/O Board from the Adapter Block Dialog Box

You can select your board by clicking on the appropriate driver. See Table 1-2 on page 1-13 for a list of all supported boards. After you've selected a board, a

dialog box for setting parameters opens. The name of this dialog block depends on which board you've installed. An example of the dialog box is shown below.

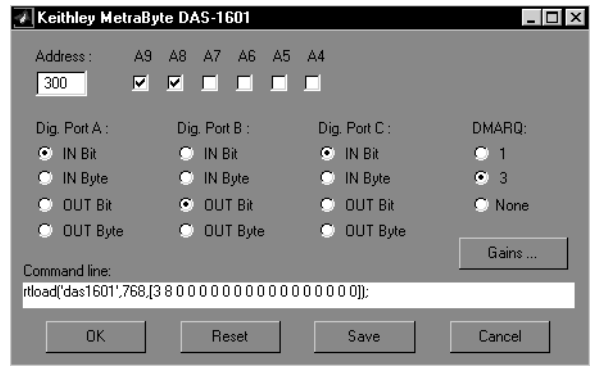


Figure 1-6: Parameter Settings Dialog Box of the Adapter Block

To show how the Adapter block works in a model, see `rtsignal` below, one of the demos provided with the Real-Time Windows Target.

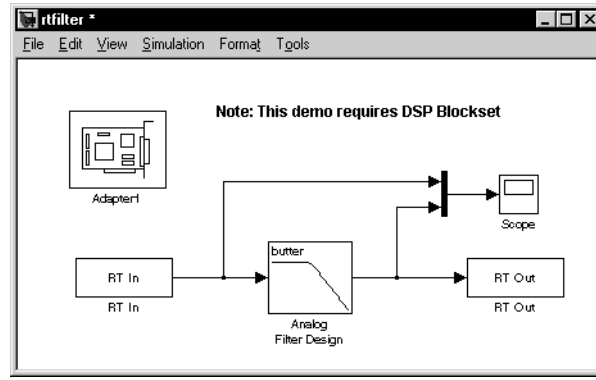


Figure 1-7: A/D Converter via RT In, and D/A Converter via RT Out

The RT In and RT Out blocks provide a gateway to the physical I/O board and indicate signal connectivity so that Real-Time Workshop's generated code correctly maps block diagram signals from or to the appropriate I/O channel(s). Accordingly, RT In and RT Out work as source and sink blocks. The difference is that these blocks exchange signals with physical devices such as D/A and A/D converters. The dialog box of each RT In and RT Out block provides an

entry for the particular Adapter block that it is associated with. For example, you could set up Adapter1 to reference a DAS1601 board. Then, to read one or more A/D signals from the DAS1601, place an RT In block in your block diagram and specify in its dialog box that it uses Adapter1. RT Out blocks are configured in a similar way.

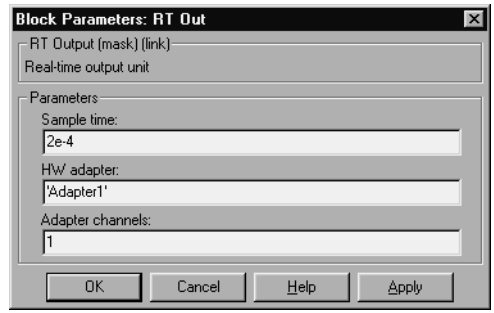


Figure 1-8: Dialog Box for RT In Block Using Adapter1

I/O Board Support

The Real-Time Windows Target includes support for more than 60 I/O boards. Multiple boards may be used as I/O for a model provided they have nonoverlapping base addresses. If you have a board that is not listed here, you can support it by adding your own I/O driver. See "Targeting DOS for Real-Time Applications" in the *Real-Time Workshop User's Guide* for more information.

Compatible I/O Boards

This table lists the I/O boards supported by the Real-Time Windows Target.

Table 1-2: Compatible DAQ Boards

Manufacturer	Board Name	Driver Name
Advantech	PCL-1800	`pcl1800`
	PCL-711B	`pcl711b`
	PCL-726	`pcl726`
	PCL-727	`pcl727`
	PCL-728	`pcl728`
	PCL-812	`pcl812`
	PCL-812PG	`pcl812pg`
	PCL-814B	`pcl814b`
	PCL-816	`pcl816`
	PCL-818	`pcl818`
	PCL-818H	`pcl818h`
	PCL-818HD	`pcl818hd`
	PCL-818HG	`pcl818hg`
	PCL-818L	`pcl818l`

Table 1-2: Compatible DAQ Boards (Continued)

Manufacturer	Board Name	Driver Name
Analog Devices	RTI-800	`rti800`
	RTI-800-A	`rti800a`
	RTI-800-F	`rti800f`
	RTI-815	`rti815`
	RTI-815-A	`rti815a`
	RTI-815-F	`rti815f`
	RTI-2100-LC	`rti210lc`
	RTI-2100-D	`rti210d`
	RTI-2100-DA	`rti210da`
	RTI-2100-DS	`rti210ds`
	RTI-2100-DS4	`rti210d4`
	RTI-2100-PGH	`rti210gh`
	RTI-2100-PGL	`rti210gl`
	RTI-2100-PGSH	`rti210sh`
	RTI-2100-PGSL	`rti210sl`
Axiom	AX5412-H	`ax5412h`
	AX5412-L	`ax5412l`
	AX5611C	`ax5611c`
Computer Boards	CIO-DAS08/Jr	`cdas08jr`
	CIO-DAS08/Jr-AO	`cdas08ja`
	CIO-DAS-1601/12	`cdas1601`

Table 1-2: Compatible DAQ Boards (Continued)

Manufacturer	Board Name	Driver Name
Computer Boards (continued)	CIO-DAS-1602/12	`cdas1602`
	CIO-DAS-1602/16	`cdas1606`
Data Translation	DT2801	`dt2801`
	DT2801-A	`dt2801a`
	DT2801/5716	`dt280116`
	DT2805	`dt2805`
	DT2805/5716	`dt280516`
	DT2811-PGH	`dt2811h`
	DT2811-PGL	`dt2811l`
Humusoft	AD1216	`ad1216`
	AD512	`ad512`
Keithley-Metrabyte	ADC-16	`adc16`
	DAS-1401	`das1401`
	DAS-1402	`das1402`
	DAS-8	`das8`
	DAS-8/AO	`das8ao`
	DAS-8PGA	`das8pga`
	DAS-16G1	`das16g1`
	DAS-16G2	`das16g2`
	DAS-1601	`das1601`
	DAS-1602	`das1602`

Table 1-2: Compatible DAQ Boards (Continued)

Manufacturer	Board Name	Driver Name
Keithley-Metrabyte (continued)	DDA-06	`dda06`
National Instruments	AT-MIO-16E-1	`atm16e1`
	AT-MIO-16E-2	`atm16e2`
	AT-MIO-64E-3	`atm64e3`
	AT-MIO-16E-10	`atm16e10`
	AT-MIO-16DE-10	`atm16d10`
	AT-AI-16XE-10	`ati16x10`
	AT-MIO-16XE-50	`atm16x50`
	DAQCard-1200	`daqc1200`
	Lab-PC	`labpc`
	Lab-PC+	`labpcx`
Scientific Solutions	LabMaster DMA	`labmastr`
Other	Microsoft Windows Mouse	`mousedrv`

Tutorial

This section discusses in detail how to work with the Real-Time Windows Target for Simulink. To be able to work with this product, you should first become familiar with creating models using Simulink and at least with basic concepts of Real-Time Workshop automatic code generation using the generic real-time template makefile, `grt`.

Building the Real-Time Model

This section describes how to build the real-time model.

The automatic build process is started from the Simulink model window by either clicking the build button in the **Real-Time Workshop** page of the **Simulation Parameters** dialog box or selecting **RTW Build** under the **Tools** menu. The build process consists of these steps:

1. Real-Time Workshop creates *`model_name`*`.c` and *`model_name`*`.h` C source files.
2. The `build` command for real-time applications (`make_rtw`) is issued, which creates *`model_name`*`.mk` makefile from the template makefile (`win_watc.tmf`).
3. The make utility (`wmake.exe`) builds the `model_name.rtd` real-time model using the makefile created in step 2.

The `.rtd` model is a Pharlap REX format executable. You can load an `.rtd` model to the Real-Time Windows Target kernel and execute it in real time. Refer to *The Real-Time Workshop User's Guide* for more information about the build process.

Setting Real-Time Options

Before you generate code with the Real-Time Workshop, it is necessary to set the fixed-step solver step size and to specify an appropriate fixed-step solver if the model contains any continuous-time states. At this time, you should also select an appropriate sample rate for your system. Refer to *The Real-Time Workshop User's Guide* for additional information.

The **Real-Time Workshop** page of the **Simulation Parameters** dialog box allows you to set several options for the real-time model. The following figure shows the correct Real-Time Workshop settings for using Real-Time Windows Target.

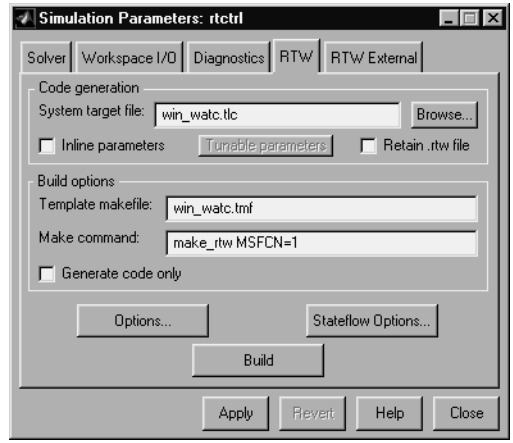


Figure 1-9: Real-Time Options Dialog Box

System Target File

You must specify the **System target file** for the Real-Time Windows Target on the **Real-Time Workshop** page of the **Simulink Parameters** dialog box as `win_watc.tlc`. We recommend that the inline parameters checkbox be left unchecked. Otherwise, this disables parameter tuning via external mode.

Template Makefile

The template makefile serves as a template for generating the real makefile, which is used by the make utility during model compilation. During the automatic build procedure, the `make_rtw` command is issued; this command extracts information from the template makefile `win_watc.tmf` and generates the template makefile `win_watc.mk`.

Set the **Template makefile** to `win_watc.tmf` for use with the Watcom C compiler (version 11.0).

Make Command

The standard `make_rtw` command supplied with the Real-Time Workshop should be used as the **Make command**. Parameter settings in this window belong to the model information; they are therefore saved with the model and kept in the model file.

Creating the Real-Time Model

You can create the real-time model itself in the same way as any other Simulink model by using standard blocks and C-MEX S-functions. You can add input and output devices in the model in several ways:

- Using the Real-Time Windows Target library with the Adapter, RT In, and RT Out blocks
- Using blocks from DOSLIB library provided with the Real-Time Workshop
- Combining both methods, that is, using both I/O device drivers from DOSLIB and the Adapter, RT In, and RT Out blocks
- Creating your own custom I/O blocks and device drivers

This manual discusses these topics in more detail in "Advanced Topics" on page 1-22 and in "Device Driver GUI Settings" on page 2-2.

Building a Model

Once you have designed the model, you can generate C code and build a real-time executable by clicking the **Build** button on the **Real-Time Workshop** page of the **Simulation Parameters** dialog box. The automatic build process creates the `.rtd` file containing a real-time model image that can run in the Real-Time Windows Target kernel.

Running the Real-Time Model

This section describes how to run the real-time model by using Simulink's external mode.

Setting External Options

Simulink's external mode is used to run the real-time model, change model parameters, and monitor model block outputs. The Real-Time Windows Target provides a communication routine, `win_tgt.dll`, that works with Simulink's external mode to support all these features. To ensure proper functionality,

this routine must be entered as the **MEX-file for external interface** in the **External Target Interface** dialog box. These are the steps:

1. Select **External Mode Control Panel** under the **Tools** menu.
2. Click the **Target Interface** button on the **External Mode Control Panel** dialog box.
3. Enter `win_tgt` as the **MEX-file for external interface** on the **External Target Interface** dialog box.
4. Click the **OK** button

This figure shows the **External Mode Control Panel**.

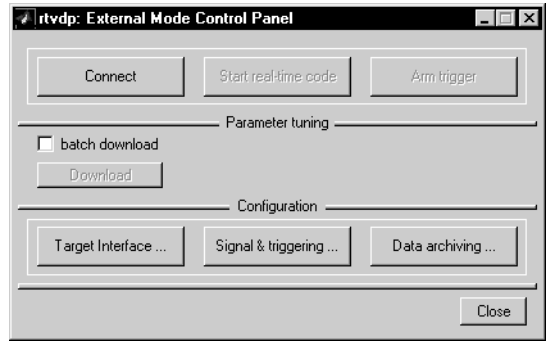


Figure 1-10: The External Mode Control Panel

This figure shows the **External Target Interface** dialog box with the correct MEX-file for external interface specification.

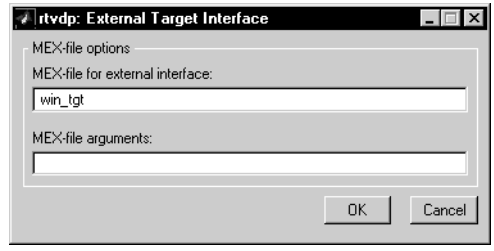


Figure 1-11: The External Target Interface Dialog Box

The real-time kernel is highly optimized for fast real-time performance under Windows 95 or Windows 98. As a result of these optimizations, data logging from the **Real-Time Workshop** page settings is disabled. Entering `win_tgt` (the default) as the **MEX-file for external interface**, however, makes signal monitoring available. Refer to the *Real-Time Workshop Users's Guide* for information on how to use Simulink's external mode.

Starting Model Execution

After setting all the options described above for the **Real-Time Workshop** page and the **External Target Interface** dialog box, you can load the model and control model execution as follows:

1. From the **Simulation** menu, select **External**.
2. From the **Simulation** menu, select **Connect to target**.
3. From the **Simulation** menu, select **Start real-time code**.
4. If you want to stop model execution, select **Stop real-time code** item under **Simulation** menu.

Changing Model Parameters

You must use Simulink's external mode to change model parameters. While external mode is running, you can open any Simulink block and change a parameter value. External mode will automatically transfer the new value to the real-time executable during execution. When a parameter in a Simulink model is changed, the communication module `win_tgt.dll` transfers the data to the external real-time model 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 avoid conflicts.

Logging Model Outputs

You can observe model outputs by using Simulink's Scope block. Data from the compiled real-time model is automatically transferred by the communication module.

Advanced Topics

Multiple I/O Boards

It is possible to use multiple hardware drivers at the same time, and hence use more than one data acquisition board at a time. You can do this by including more than one Adapter block in your Simulink model. Each RT Out and RT In block is assigned uniquely to an Adapter block. Channel numbers are independent for each adapter.

Using Custom I/O Device Drivers

Custom I/O device drivers can be used in combination with the Real-Time Windows Target. Due to the additional complexity of device drivers supplied with Real-Time Windows Target, we recommend that device drivers be written in the same style as the device drivers that are provided with the DOS target support included in with Real-Time Workshop. We do not recommend using RT In and RT Out for custom device drivers.

Source code for the DOS target device drivers is located in *matlabroot*`\rtw\c\dos\devices`. This table lists the available DOS device drivers.

Table 1-3: DOS Device Drivers Included with the Real-Time Workshop

Type of Device Driver	Filename

Digital to Analog	`das16da.c`
	`das16da.h`
	`das16da.tlc`

Analog to Digital	`das16ad.c`
	`das16ad.h`
	`das16ad.tlc`

Digital Input	`das16di.c`
	`das16di.h`
	`das16di.tlc`

Digital Output	`das16do.c`
	`das16do.h`
	`das16do/tlc`

Device drivers written in this style (that is, as inlined S-functions) are compatible with Real-Time Windows Target. They may be used in combination with the device drivers provided with Real-Time Windows Target.

See "Targeting DOS for Real-Time Applications" for more information on these I/O device drivers and "Targeting Custom Hardware" for information about writing custom device drivers; both these chapters are in the *Real-Time Workshop User's Guide*.

Detecting Excessive Sample Rates

If your specified sample rate is too fast, the Real-Time Windows Target detects and reports this fact during real-time execution. Sampling rates up to 30 kHz can be achieved on Pentium computers. Once the model is running, the `rtwho` command can be issued in the MATLAB command line to observe the system performance. As indicated, MATLAB performance decreases as the system becomes overloaded.

```
Real Time Windows Target version 1.00 (C) The MathWorks, Inc.
1998
MATLAB performance = 46.0%
Kernel timeslice period = 0.03 ms

TIMERS:  Number      Type     Period   Running

            1          In   0.00003      Yes

HISTORIES:      Name         Size      Mode      Link

        _history_ext        1 by 1       0

DRIVERS:                          Name   Address   Inputs  Outputs

                 Microsoft Windows Mouse         0      2       0
                        Real-Time Model          0      4       0
```

Limitations

The following blocks are not supported by the Real-Time Windows Target:

- XY Graph
- To Workspace

- To File
- Stop Simulation
- Display

The DSP Blockset is compatible with Real-Time Windows Target. The Power Systems Blockset, Fuzzy Logic Toolbox, Communications Toolbox, and Fixed-Point Blockset are not currently supported with Real-Time Windows Target.

2

Device Driver Settings

Device Driver GUI Settings

The Real-Time Windows Target provides more than 60 I/O device drivers for interfacing your block diagram to real-world sensors and actuators. I/O boards are connected to the backplane of your PC. By reading and writing data to special memory locations, the Real-Time Windows Target reads from and writes to the board. The board automatically converts from electrical signals (analog or digital) to a digital word that is written to or read from special memory locations.

The I/O device drivers provided enable the Real-Time Windows Target to transfer series of data values to or from the real-time version of your Simulink block diagram. Typically I/O boards are preset from the factory for certain base addresses, voltage levels, and unipolar (e.g. 0 to 5 volts, or 0 to 10 volts) or bipolar (e.g. +/- 5 volts, or +/- 10 volts) modes of operation. Boards also include switches or jumpers that allow you to change many of these initial settings. For information regarding setup and installation of any I/O board, please read the board manufacturer's documentation.

The Real-Time Windows Target provides flexibility similar to that offered by board manufacturers. To provide flexibility for changing the base addresses, voltage levels, and unipolar or bipolar modes, you must make the same setting information available to the Real-Time Windows Target. This is done by user entries in the I/O device driver graphical user interfaces (GUIs). This is necessary because most manufacturers do not provide a means for the I/O device driver to read all board settings programmatically.

Setting the Base Address

Most I/O boards are preset with the base address of `0x300`. If you are using multiple I/O boards or other boards (network cards, etc.) that already occupy the address `0x300`, you must set up your board with another base address and also replicate the settings in the I/O device driver GUIs in your block diagrams. See your board manufacturer's user's manual for board settings. The Real-Time Windows Target provides a dialog box entry for declaring the base address that you have set. Match the base address switch or jumper setting on the I/O board. This dialog box is declared in the Adapter block GUI. Once you have declared the base address in the adapter block, you then close the dialog box. At this time, Real-Time Windows Target checks the specified base address and attempts to search for the board at the base address that you've specified. If Real-Time Windows Target confirms that the selected board is located at

that base address, then when you close the dialog box, the border color of an Adapter block will be changed to black, indicating that configuration was correct. If the board has not been configured or the specified board cannot be found, the Adapter block border color will remain red. Once you have selected a device driver by name in the Adapter block, the Adapter block I/O device driver and the base address cannot be altered. To use a different I/O board device driver or to choose a new base address, it is necessary to delete the existing Adapter block, copy a new Adapter block in from the `rtwinlib` library, and declare the appropriate settings for this new Adapter block.

Channel Selection

Real-Time Windows Target uses the RT In block for all input signals: analog inputs, digital inputs, and digital words. The RT Out block handles all output signals: analog outputs, digital outputs, and digital words. For a better understanding of how to specify device driver settings when using both digital and analog signals, this section illustrates how this is done with examples based on the Keithley-Metrabyte DAS-1601 board. The following is a summary of relevant specifications of the DAS-1601 board:

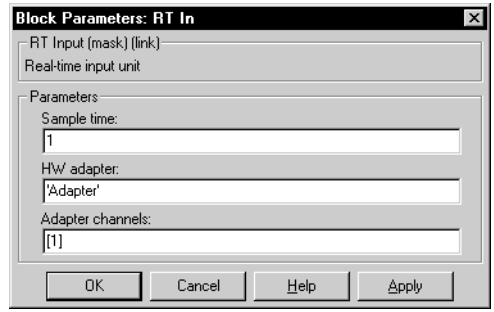
- 16 single-ended or 8 differential analog inputs, switch configurable as either unipolar (0 to 10 volts) or bipolar (+/- 10 volts)
- Switch configurable base address
- Programmable gains of 1,10,100, and 500
- Four unidirectional digital inputs
- Four unidirectional digital outputs
- Two 12-bit digital-to-analog converter channels, switch configurable as 0 to 5 volts, 0 to 10 volts, +/- 5 volts, or +/- 10 volts
- An additional 24 bits of bidirectional digital I/O from the PIO connector (J2)

We will explore several different configurations for input signals.

Once the Adapter block has been placed in the model and given the correct base address (one that does not conflict with any other hardware), you can set up the RT In block to handle input signals. The most basic case is for a single analog input signal that will be physically connected to the first analog input signal on the board. In the RT In block's GUI, you should set the entry for channels to:

```
[ 1 ]
```

The use of brackets is optional. This figure shows the RT In block's dialog box with the correct settings.



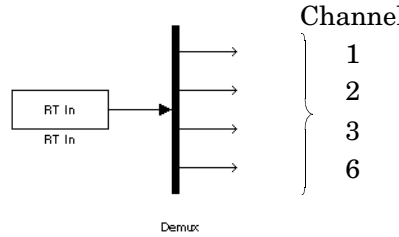
Now assume that you want to use channels 1 through 3 as well as channel 6. All of these channels are analog. Real-Time Windows Target RT In and RT Out blocks begin channel numbering with analog signals, then conclude with digital signals. Since the DAS 1601 has 8 differential inputs or 16 single-ended inputs, signals 1 through 6 are all analog channels. In order to use only the four channels specified above, specify the channel entry in the dialog box as

```
[1:3, 6]
```

or

```
[1,2,3,6]
```

This illustration shows the resulting block diagram.



It is acceptable to specify more channels than are actually used. Generally, this is not recommended, however, because it results in additional overhead for the processor and A/D or D/A converters. In this case, even though some channels are not actually used in the block diagram, these channels are still converted.

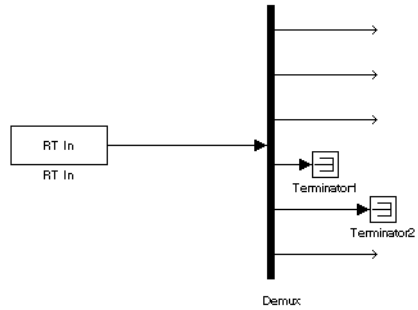
In the example described above, if you decided to accept the additional conversion overhead, you could specify the channels as

```
[1,2,3,4,5,6]
```

or

```
[1:6]
```

Then, you could simply attach terminator blocks to channels 4 and 5 inside your block diagram after passing the RT In block vector into a Demux block. This illustration 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.

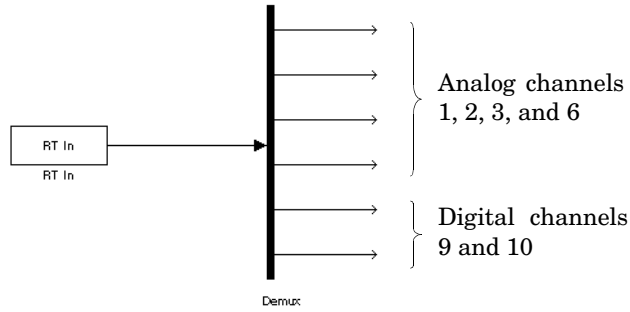
In the next example, assume that you now want the first digital input signal only. In a contiguous vector, digital input signals follow the analog input signals. Assuming that you are using differential analog inputs, which means that there are a total of eight analog inputs, your first digital input is the ninth element from the RT In block. Digital input channels are numbered 9 through 16. To input only the first digital input signal, provide the following entry:

```
[9]
```

To use both analog and digital inputs, consider an example where you would like to input analog signals 1,2,3 and 6. To extend this example to also use the first two channels of digital input, the correct dialog box entry is

```
[1,2,3,6,9,10]
```

This illustration shows the block diagram implementation.



Note that, if the board is configured with 16 single-ended analog inputs, then the first 16 channels would be designated as analog input channels.

3

Parameter Tuning and External Mode

External Mode Operation

Simulink's external mode provides a universal interface for use with Real-Time Workshop targets. It allows you to make parameter changes in the Simulink block diagram and have the new parameter changes be exported automatically to the real-time executable. You can use Simulink's external mode to change parameters of your real-time model on-the-fly.

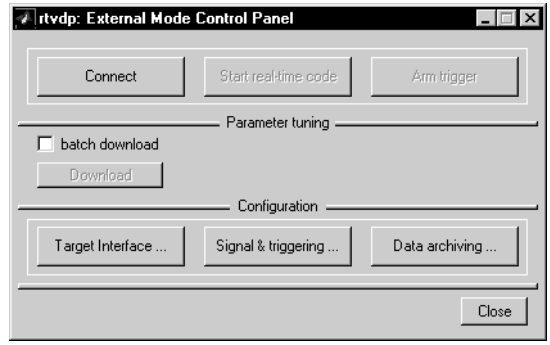
External mode requires a communications interface to pass parameters external to Simulink, and on the receiving end, the same communications protocol must be used to accept new parameter values and insert them in the proper memory locations for use by the real-time model. In some Real-Time Workshop targets 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 model.

As a user of Real-Time Windows Target, you will find that the requirements for setup are minimal. You start by enabling external mode and specifying the correct name for the MEX-file interface. Then, after you have built the real-time executable, you are ready for external mode operation. For information on how to build an external mode enabled executable, see the *Real-Time Workshop User's Guide*.

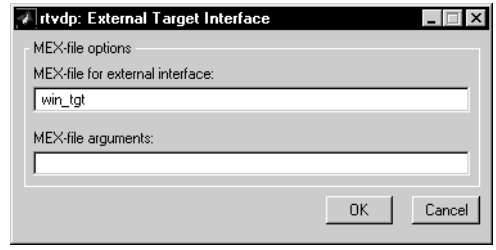
The MEX-file interface code is provided with Real-Time Windows Target and allows rapid and convenient exchange of data between the MATLAB and Simulink environment and the real-time executable during real time execution. The required **MEX-file for external interface**, `win_tgt`, should be specified on the **External Target Interface** dialog box. You can open the **External Target Interface** dialog box by following these steps:

- Under the **Tools** menu, check **External Mode Control Panel**. This opens the **External Mode Control Panel** dialog box.
- Click the **Target Interface** on the **External Mode Control Panel**. This opens the **External Target Interface** dialog box.

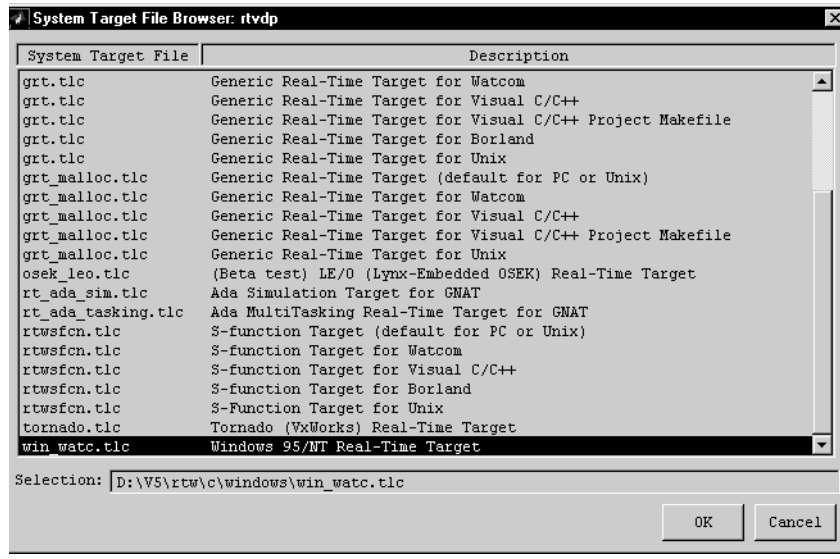
This picture shows the **External Mode Control Panel**:



This picture shows the **External Target Interface** dialog box with the correct settings:



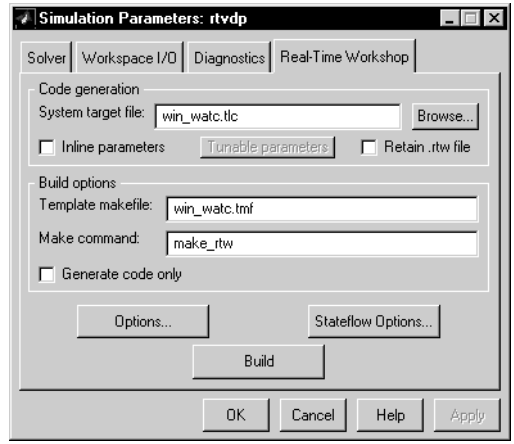
Once you have correctly set the **External Target Interface**, open the **Real-Time Workshop** page by selecting **RTW Options** under the **Tools** menu. You can specify the correct system target file, template makefile, and make command by clicking the **Browse** button, which opens the System Target File Browser:



Select `win_wat.tlc` as the system target file and click **OK**. This specifies these three files:

- **System target file** — `win_wat.tlc`
- **Template makefile** — `win_wat.tmf`
- **Make command** — `make_rtw`

The picture below shows the correct settings for the **Real-Time Workshop** page.



Do not check **Inline parameters** on the **Real-Time Workshop** page of the **Simulation Parameters** dialog box when using the Real-Time Windows Target. With external mode specified in the dialog boxes as shown above, you can select **Build** to generate your real-time executable. This mode of operation makes it possible to tune parameters during real-time execution and to view signals from the real-time model using Simulink Scope blocks. These two capabilities are described in greater depth in the following sections.

Parameter Tuning

Parameter tuning with Real-Time Windows Target is effortless. Once code is generated and built, you can change parameter values by simply editing a block diagram and changing block parameters in a dialog box; the new value(s) is automatically downloaded to the real-time executable. If you have changed a parameter value, it is exported to the real-time model immediately after you have completed any one of these tasks:

- Select **Apply**
- Select **OK**
- Press carriage return

There is a slight difference when tuning MATLAB variables. If a block parameter has been entered as a MATLAB variable, the value of the parameter will be exported only when you perform one of the tasks listed above or by selecting **Update Diagram** under the **Edit** menu. 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 model during execution.

Capturing and Displaying Signals

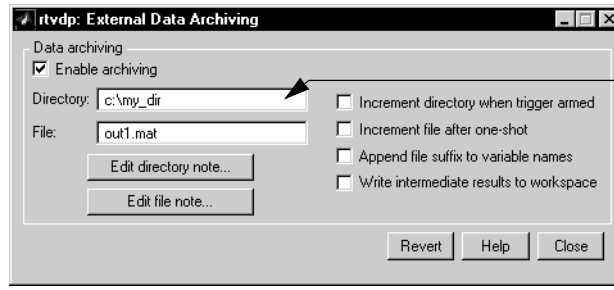
Integration between Simulink's external mode and the Real-Time Windows Target enables you to capture and display signals (i.e., the wires within a Simulink block diagram) from the real-time model. During real-time execution, Simulink's external mode provides bidirectional exchange of data. New model parameter values can be passed down to the real-time model while signal data can be retrieved from the real-time model and displayed in Simulink Scope blocks.

As in parameter tuning, capture and display of signals requires that you have set `win_tgt` as the **MEX-file for external interface** in the **External Target Interface** page.

You can monitor any signal in your Simulink block diagram if the signal is connected to a Scope block before code generation. For signal monitoring to start, you must make sure that the following conditions have been met:

- External mode enabled prior to code generation
- `win_tgt` MEX-file interface specified in the **External Target Interface** page
- Executable built
- Signal selected under data archiving configuration
- Signal armed
- Connect to target
- Start real-time
- Sufficient residual time between samples (e.g., idle CPU time).

From the **External Mode Control Panel**, press the **Data Archiving** button. This opens the **External Data Archiving** dialog box. This graphical user interface (GUI) allows you to specify data archiving options.



Specify the directory and file-name for your archived data. For information about other options, see the *Real-Time Workshop User's Guide*.

At each sample interval of the real-time model, Simulink stores contiguous data points in memory until filling a data buffer. Once the data buffer is filled, Simulink suspends data capture while the data is transferred back to MATLAB via Simulink external mode. Your model, however, continues to run. Transfer of data is less critical than maintaining deterministic real-time model 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 Simulink's external mode.

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

To set up data collection, click the **Signal & Triggering** button of the **External Mode Control Panel** and then provide the desired settings in the **External Signal & Triggering** dialog box as shown below.

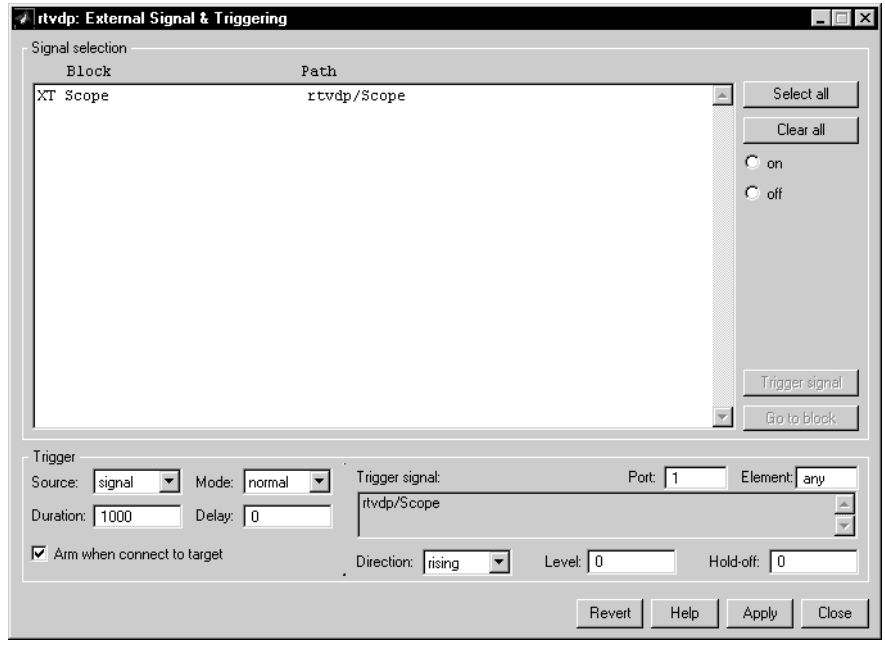


Figure 3-1: The External Signal & Triggering Dialog Box

In this GUI, 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. The duration specifies the number of contiguous points of data to be collected in each buffer of data. We 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. For a complete description of data logging configuration, please refer to the *Real-Time Workshop User's Guide*.

4

Real-Time Windows Target Troubleshooting

Troubleshooting

This chapter discusses some common errors you can encounter when using the Real-Time Windows Target.

Plots Not Visible in Simulink Scope Block

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

- `win_tgt` as the **MEX-File for external interface** on the **External Target Interface** dialog box
- **external** selected from the **Simulation** menu
- **connect to target** from the **Simulation** menu
- **External Mode Logging** under the **Tools** menu
- After clicking **configure**, select one or more signals for capture (designated with "X") in the **External Data Logging Configuration** dialog box
- 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
- **arm when connect to target** in the **External Data Logging Configuration** dialog box or **arm** in the **External Data Logging Control Panel**
- **start real-time** 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 carefully selected, it is possible that failure to obtain time responses in Scope blocks is due to insufficient CPU time available. 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 may 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 does not fall below 50%.

Compiler Error Message

If, during the build phase of your model, you see

```
incorrect version of Watcom compiler installed (version 11.0 is
required)
```

or

```
incorrect compiler installation
```

you must check **DOS target** in addition to the Windows target during installation of the Watcom 11.0 compiler.

Failure to Connect to Target

If you see

```
Checksum mismatch error
```

in the **Simulation Errors** dialog, this indicates that the model structure has changed since last time code was generated. You must rebuild the real-time executable. If your model fails to successfully build, we recommend that you delete `.mk` and `.obj` files and select **Build**.

Error Message When Closing Adapter Dialog Box

An error message will occur if the I/O board specified in the Adapter dialog box cannot be found. When the Adapter dialog is closed, the Real-Time Windows Target runs a test to determine whether it can locate the specified board at the specified I/O base address. If you have specified the correct I/O board, check that the base address setting matches the jumper settings on the I/O board and that the I/O address range does not overlap the I/O range of any other board.

Verifying the correct setting may require powering down your computer, removing the I/O board, and consulting the board vendor's documentation to confirm jumper or switch settings on the I/O board. Also check to be sure that the board is properly and securely seated in the slot.

Sample Time Too Fast

Real-Time Windows Target displays the message

```
??? RTTOOL: Too fast for this hardware
```

when it detects a failure to complete the execution of the model successfully within the specified sample interval. For hardware and model complexity, the sample interval is too small and you should select a slower sample interval. You must rebuild the real-time model after changing the sample interval.

It is possible to run a model close to the maximum limit without seeing this message. In such cases, you may notice either slow updates of Simulink Scope blocks or a complete failure to plot data in the Scope blocks. Similarly, this may be an indication that you are reaching the upper threshold for sample rate for running your model on your hardware. Plotting data is a lower priority than real-time execution of your model.

5

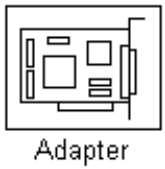
Real-Time Windows Target Block Reference

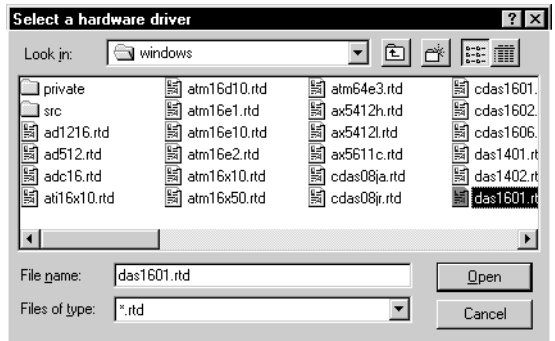
Adapter

Purpose

Select and load the Real Time Windows Target I/O board device driver

Description





The Adapter block allows you to select a driver from a library with more than 60 I/O device drivers. The placement of one Adapter block in your Simulink block diagram corresponds to one I/O board that has been installed in your PC backplane. Multiple Adapter blocks can be used in a Simulink block diagram provided that they are uniquely assigned to different I/O boards in the PC backplane.

The Adapter block located in the `rtwinlib` library is unassigned, meaning that a specific I/O board and board address have not been specified. Therefore, the Adapter block in the library will appear red in color. Similarly, an unassigned Adapter block placed in your Simulink block diagram will also be colored red. This signifies that it is uninitialized and no I/O device driver has been assigned.

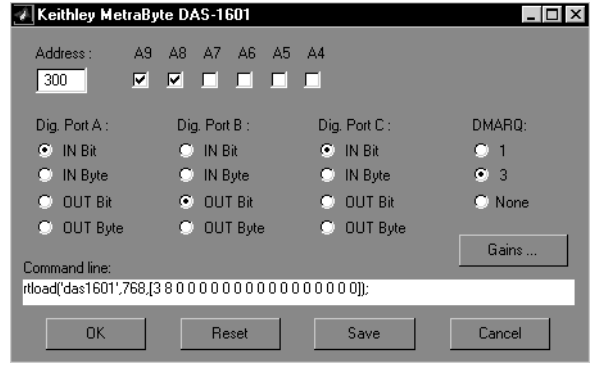
Double clicking on the Adapter block opens the **Select a Hardware Driver** dialog box. This dialog box contains the full library of I/O device drivers. You can select exactly one I/O device driver for each Adapter block. After a particular I/O device driver has been selected, a new dialog box appears that allows you to specify I/O device driver parameters including base address and the I/O type (byte, bit, etc.). Each device driver has a unique dialog box associated with it that allows the specification of parameters, including whether the board supports digital or analog inputs and outputs, and so on.

Once a base address is provided and the dialog box is closed, Real-Time Windows Target automatically tests to determine whether the specified board

can be detected at the specified address. If the results of the automatic test to confirm that valid board and board address are detected, the Adapter block will then be colored black indicating that it is correctly initialized.

Upon re-opening the Adapter block, the device driver parameters dialog box opens, skipping the step for selecting a particular I/O board and base address. Once an Adapter block is initialized, the board type and base address cannot be altered. To change the I/O device driver or other settings, you must delete the Adapter block and replace it with a new Adapter block from the I/O device driver library. When using multiple I/O boards, each board must have a unique and valid base address that does not conflict with other devices (e.g., sound cards or network cards).

Once configured, an Adapter block is uniquely associated with a particular I/O board located in your PC backplane. In combination with an Adapter block, you can place input blocks (RT In) and output blocks (RT Out) in your Simulink block diagram. Each RT In or RT Out block provides a dialog entry allowing you to specify the name of the Adapter block that determines which I/O board will be used for a particular RT In or RT Out block. Selection of particular channels of a board is performed from the RT In or RT Out dialog box.



RT Kernel Resources Used

One HW driver

See Also

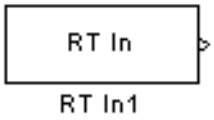
RT In, RT Out

RT In

Purpose

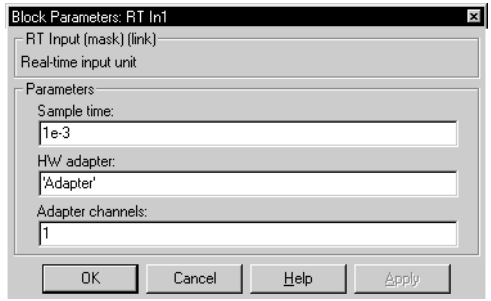
Real-time input block

Description



The RT In block is designed to capture one sample per channel at each sample interval. You can specify **Sample time** as a scalar or as a two-element vector of sample time and offset. **HW adapter** is the name of Adapter block, representing the I/O device driver card used as an input device. You can specify **Adapter channels** as either single channel number or a vector of channel numbers.

Dialog Box



RT Kernel Resources Used

One timer

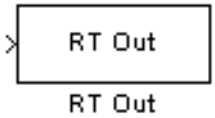
See Also

RT Out

Purpose

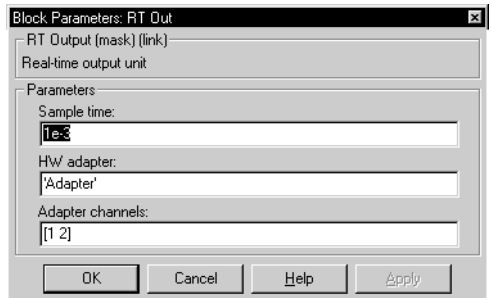
Real-time output block

Description



The RT Out block is used to output a signal to an I/O board without any buffering delay. You can specify **Sample time** as a scalar or as a two-element vector of sample time and offset. **HW adapter** is the name of Adapter block, representing the particular I/O board being used as an output device. You can specify **Adapter channels** as either a single channel number or a vector of channel numbers.

Dialog Box



RT Kernel Resources Used

One timer

See Also

RT In

6

Real-Time Windows Target Function Reference

rtclear

Purpose

Clear all defined timers and history variables

Syntax

```
rtclear
rtclear all
```

Description

`rtclear` and `rtclear all` clear all objects in the RT kernel and stop all activities. Note that `rtclear` and `rtclear all` do not remove drivers or model code.

Caution should be used if attempting to perform an emergency stop using the `rtclear` function. If `rtclear` is used as an emergency stop, you should also use `rtunload` to ensure that I/O device outputs are reset to their initial states.

See Also

`rtunload`

Purpose

Load hardware drivers

Syntax

```
rtload
rtload('driver')
rtload('driver',ADDR)
rtload('driver',ADDR,PARM)
```

Description

`rtload` displays the GUI for loading hardware drivers.

`rtload('driver')` displays a GUI for a specific driver.

`rtload('driver',ADDR)` loads a hardware driver with a specific I/O address.

`rtload('driver',ADDR,PARM)` loads a hardware driver with a specific I/O address and parameters.

Default I/O address and parameters can be changed by the **Save** button in the driver GUI.

This function is ordinarily called from external mode, but you can use this function directly from the MATLAB prompt to load your device driver manually.

See Also

`rtunload`

rtunload

Purpose Unloads all hardware drivers loaded with `rtload`

Syntax `rtunload`

Description `rtunload` unloads all hardware drivers loaded with `rtload`. You can run this function directly from the command line to unload all device drivers.

See Also `rtload`, `rtclear`

Purpose

Lists information about hardware drivers, timers, and history variables

Syntax

```
rtwho
```

Description

`rtwho` prints all useful information about the Real-Time Windows Target status, and also lists all hardware drivers, kernel performance, and the version number.

Index

T

U

W