

xPC Target

For Use with Real-Time Workshop[®]

Modeling
└─

Simulation
└─

Implementation
└─



User's Guide

Version 1.1

How to Contact The MathWorks:



508-647-7000

Phone



508-647-7001

Fax



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

Mail



<http://www.mathworks.com>
<ftp.mathworks.com>
<comp.soft-sys.matlab>

Web
Anonymous FTP server
Newsgroup



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

Technical support
Product enhancement suggestions
Bug reports
Documentation error reports
Subscribing user registration
Order status, license renewals, passcodes
Sales, pricing, and general information

xPC Target User's Guide

© COPYRIGHT 2000 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 or for the federal government of the United States. By accepting delivery of the Program, the government hereby agrees that this software qualifies as "commercial" computer software within the meaning of FAR Part 12.212, DFARS Part 227.7202-1, DFARS Part 227.7202-3, DFARS Part 252.227-7013, and DFARS Part 252.227-7014. The terms and conditions of The MathWorks, Inc. Software License Agreement shall pertain to the government's use and disclosure of the Program and Documentation, and shall supersede any conflicting contractual terms or conditions. If this license fails to meet the government's minimum needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to MathWorks.

MATLAB, Simulink, Stateflow, Handle Graphics, and Real-Time Workshop are registered trademarks, and Target Language Compiler is a trademark of The MathWorks, Inc.

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

Printing History: September 1999 First printing New for version 1.0 (Release 11.1)
November 2000 Second printing Revised for version 1.1 (Release 12.0)

Preface

Documentation	xi
Online Documentation	xi
Printing the Documentation	xi
Required Products	xii
MATLAB	xii
Simulink	xiii
Real-Time Workshop	xiv
C Compiler	xiv
Related Products	xv
Stateflow	xv
Stateflow Coder	xv
DSP Blockset	xvi
Dials & Gauges Blockset	xvi
Using This Guide	xvii
Expected Background	xvii
Organization	xviii
Conventions	xix
Terminology	xix
Typographical	xxi

What Is xPC Target?	1-3
Features of xPC Target	1-4
Real-Time Kernel	1-4
Real-Time Application	1-6
Signal Acquisition and Analysis	1-6
Parameter Tuning	1-7
Hardware Environment	1-8
Host PC	1-8
Target PC	1-8
Host-Target Connection	1-9
Input/Output Driver Support	1-11
Software Environment	1-12
Host-Target Communication	1-12
Rapid Prototyping Process	1-13
Embedded Process	1-15
User Interaction	1-18
xPC Target Graphical Interface	1-19
MATLAB Command Line Interface	1-20
Target PC Command Line Interface	1-21
Simulink External Mode Interface	1-21
Simulink Dials and Gauges Interface	1-22
Web Browser Interface	1-22

System Requirements	2-3
Host PC	2-3
Target PC	2-4
Installation on the Host PC	2-7
Getting or Updating Your License	2-7
CD-ROM Installation	2-8
Web Download Installation	2-8
Files on the Host PC Computer	2-10
Initial Working Directory	2-11
Setting Your Working Directory from the Desktop Icon	2-11
Setting Your Working Directory from Within MATLAB	2-11
Serial Communication	2-12
Hardware for Serial Communication	2-12
Environment Properties for Serial Communication	2-13
Network Communication	2-15
Hardware for Network Communication	2-16
Ethernet Card for a PCI-Bus	2-18
Ethernet Card for an ISA-Bus	2-18
Environment Properties for Network Communication	2-20
Target Boot Disk	2-23
Current Properties on the Target Boot Disk	2-25
Testing and Troubleshooting the Installation	2-26
Testing the Installation	2-26
Test 1, Ping Target System Standard Ping	2-27
Test 2, Ping Target System xPC Target Ping	2-29
Test 3, Reboot Target Using Direct Call	2-30
Test 4, Build and Download Application	2-30
If You Still Need More Help	2-31

Simulating the Model	3-3
Loading a Simulink Model	3-3
Running a Simulation Using the Simulink Graphical Interface	3-4
Running a Simulation Using the MATLAB Command Line Interface	3-5
Creating the Target Application	3-7
Booting the Target PC	3-7
Troubleshooting the Boot Process	3-8
Entering the Simulation Parameters	3-8
Building and Downloading the Target Application	3-13
Troubleshooting the Build Process	3-15
Controlling the Target Application	3-16
Control with MATLAB Commands	3-17
Signal Monitoring	3-19
Signal Monitoring with MATLAB Commands	3-19
Signal Logging	3-20
Signal Logging with xPC Target Graphical Interface	3-20
Signal Logging with MATLAB Commands	3-22
Signal Tracing	3-26
Signal Tracing with xPC Target GUI	3-26
Signal Tracing with xPC Target GUI (Target Manager)	3-31
Signal Tracing with MATLAB Commands	3-35
Parameter Tuning	3-38
Parameter Tuning with MATLAB Commands	3-38
Parameter Tuning with Simulink External Mode	3-40

I/O Driver Blocks	4-3
xPC Target I/O Driver Blocks	4-3
Adding I/O Blocks with the xPC Target Library	4-4
Adding I/O Blocks with the Simulink Library Browser	4-7
Defining I/O Block Parameters	4-10
xPC Target Scope Blocks	4-13
xPC Target Scope Blocks	4-13
Adding xPC Target Scope Blocks	4-14
Defining xPC Target Scope Block Parameters	4-16
Target PC Command Line Interface	4-19
Using Methods and Properties on the Target PC	4-19
Target Object Methods	4-20
Target Object Properties	4-20
Scope Object Methods	4-21
Scope Object Properties	4-22
Using Variables on the Target PC	4-24
Variable Commands	4-24
Web Interface	4-25
Connecting the Web Interface	4-25
Using the Main Page	4-26
Changing WWW Properties	4-28
Viewing Signals with the Web Browser	4-28
Using Scopes with the Web Browser	4-29
Viewing and Changing Parameters with the Web Interface	4-30
Changing Access Levels to the Web Browser	4-31

Introduction	10-3
DOSLoader Mode Overview	10-4
StandAlone Mode Overview	10-4
Architecture	10-5
Restrictions	10-6
 Updating the xPC Target Environment	10-8
 Creating a DOS System Disk	10-11
 DOS Loader Target Applications	10-12
Creating a Target Boot Disk for DOS Loader	10-12
Creating a Target Application for DOS Loader	10-13
 Stand-Alone Target Applications	10-14
Creating a Target Application for Stand-Alone	10-14
Creating a Target Boot Disk for Stand-Alone	10-15
Using Target Scope Blocks with Stand-Alone	10-15

Environment Reference

Environment	5-3
Environment Properties	5-3
Environment Functions	5-11
 Using Environment Properties and Functions	5-12
Getting a List of Environment Properties	5-12
Saving and Loading the Environment	5-13
Changing Environment Properties with Graphical Interface .	5-14
Changing Environment Properties with Command Line Interface	5-16
Creating a Target Boot Disk with Graphical Interface	5-17
Creating a Target Boot Disk with Command Line Interface .	5-19

System Functions	5-20
GUI Functions	5-20
Test Functions	5-21
xPC Target Demos	5-21
 Environment and System Function Reference	 5-23

7 | **Target Object Reference**

Target Object	6-3
What is a Target Object?	6-3
Target Object Properties	6-4
Target Object Methods	6-9
Target PC Commands	6-11
 Using Target Objects	 6-13
Displaying Target Object Properties	6-13
Setting the Value of a Target Object Property from the Host PC	6-14
Setting the Value of a Target Object Property from the Target PC	6-15
Getting the Value of a Target Object Property	6-16
Using the Method Syntax with Target Objects	6-17

Scope Object 7-3

 What is a Scope Object? 7-3

 Scope Object Properties 7-3

 Scope Object Methods 7-6

Using Scope Objects 7-8

 Displaying Scope Object Properties for a Single Scope 7-8

 Displaying Scope Object Properties for All Scopes 7-9

 Setting the Value of a Scope Property 7-9

 Getting the Value of a Scope Property 7-10

 Using the Method Syntax with Scope Objects 7-11

Preface

Documentation xi
Online Documentation xi
Printing the Documentation xi
 Required Productsxii
MATLAB xii
Simulink xiii
Real-Time Workshop xiv
C Compiler xiv
 Related Products xv
Stateflow xv
Stateflow Coder xv
DSP Blockset xvi
Dials & Gauges Blockset xvi
 Using This Guide xvii
Expected Background xvii
Organization xviii
 Conventions xix
Terminology xix
Typographical xxi

xPC Target is part of a family of software products that you use to create real-time control systems. Some of these products are required while others you use for special applications.

This chapter includes the following sections:

- **“Documentation”** - Online and PDF files
- **“Documentation”** - MATLAB[®], Simulink[®], Real-Time Workshop[®], xPC Target, and a C compiler
- **“Related Products”** - Stateflow[®], Stateflow Coder, DSP Blockset, Dials & Gauges Blockset, xPC Target Embedded Option, and Web browser
- **“Using This Guide”** - Suggestions for learning about xPC Target, finding information, and a description of the chapters
- **“Conventions”** - Terms that may have various meanings and text formats in this guide

Documentation

The xPC Target software is shipped with a *Getting Started Guide*. The remaining documentation is available online through the Help browser window, or as PDF files that you can view online or print.

This section includes the following topics:

- “Online Documentation”
- “Printing the Documentation”

Online Documentation

Access to the online information for xPC Target is through the MATLAB window.

- 1 In the MATLAB window, from the **View** menu, click **Help**.

The Help browser window opens.

- 2 In the left pane, click **xPC Target**.

The Help browser displays the xPC Target Roadmap page in the right pane. Access the online information by selecting the links under the “Learning About xPC Target” section.

Alternatively, you can access the online information by using the table of contents in the left pane. Click the + and - boxes to display and hide hierarchical levels.

Printing the Documentation

Access to the PDF files for xPC Target is through the xPC Target Roadmap page or from the pdf_doc directory. The locations listed below assume you have installed the files from the documentation CD.

The following manuals are available as PDF files.

- **xPC Target User’s Guide** - located at
C:\MATLAB root\help\pdf_doc\xpc\xpc_target_ug.pdf
- **xPC Target I/O Reference Guide** - located at
C:\MATLAB root\help\pdf_doc\xpc\xpc_i_o_ref.pdf

Required Products

xPC Target is a PC-compatible product, that you install on a host computer running Microsoft Windows 95, Windows 98, Windows 2000, or Window NT.

xPC Target requires the following products from The MathWorks:

- **MATLAB** - Command line interface for xPC Target
- **Simulink** - Environment to model physical systems and controllers using block diagrams
- **Real-Time Workshop** - Converts Simulink blocks and code from Stateflow Coder into C code

In addition, xPC Target requires one of the following C compilers to compile C code from Real-Time Workshop into executable code:

- **Microsoft Visual C/C++**
- **Watcom C/C++**

MATLAB

MATLAB provides a command line interface for xPC Target.

With xPC Target, you have full control of the target computer and target application using MATLAB functions and the command line interface or M-file scripts. You use the MATLAB functions for:

- **Real-time application control** - Download, start, and stop the target application
- **Signal acquisition and analysis** - Save signal data while the target application is running and analyze the data after the application has completed running, or display signal data while the target application is running in real-time
- **Parameter tuning** - Change parameters while the target application is running in real-time

Note Version 1.1 of xPC Target requires MATLAB version 6.0 on the R12 CD.

MATLAB documentation - For information on using MATLAB, see the *MATLAB Getting Started* manual. It 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 the online *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.

You can use xPC Target with most Simulink blocks including discrete-time and continuous-time systems. When you use a continuous-time system and generate code with Real-Time Workshop[®], you must use a fixed-step integration algorithm.

C-code S-functions are supported by Real-Time Workshop. However, M-code S-functions are not supported.

xPC Target I/O driver blocks - With xPC Target, you can remove the physical system model and replace it with I/O driver blocks connected to your sensors and actuators. The xPC Target I/O library supports more than 150 driver blocks, and you can download additional drivers from The MathWorks Web site.

The I/O device drivers are written as Simulink C-code S-functions.

Note Version 1.1 of xPC Target requires Simulink version 4.0 on the R12 CD.

Simulink documentation - For information on using Simulink, see the *Using Simulink* manual. It 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 a third-party C compiler, compile the code into a real-time executable.

Features of Real-Time Workshop include support for multirate systems, as well as *loop-rolling* and S-function *inlining*, which allow you to optimize your code for size and efficiency.

With xPC Target, you can build and download your target application to the target computer using the Build command in Real-Time Workshop.

Note Version 1.1 of xPC Target requires Real-Time Workshop version 4.0 and the R12 CD.

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

C Compiler

The C compiler creates executable code from the C code generated from Real-Time Workshop and the C-code S-functions you have created. This executable code is an image that is the base of the target application.

In addition to the products from The MathWorks, you need to install a C compiler. Real-Time Workshop and xPC Target support the following C compilers:

Microsoft Visual C/C++

Version 1.1 of xPC Target requires Microsoft Visual Studio C/C++ version 5.0 or 6.0.

Watcom C/C++

Version 1.1 of xPC Target requires Watcom C/C++ version 10.6 or 11.0.

Related Products

In addition to the required products from The MathWorks, the following products are compatible with xPC Target:

- **Stateflow** - Model complex systems and logic using flow and state transition diagrams.
- **Stateflow Coder** - Convert Stateflow blocks into code used by Real-Time Workshop
- **DSP Blockset** - Add digital signal processing blocks to the Simulink library.
- **Dials & Gauges Blockset** - Add graphical blocks to the Simulink library for creating an instrument control panel with a second Simulink model

Stateflow

Stateflow provides a graphical design and development tool for complex control and supervisory logic problems. It uses flow diagram notation and state transition notation to model complex system behavior.

Note Version 1.1 of xPC Target requires Stateflow version 4.0 on the R12 CD.

Stateflow documentation - For information on creating state flow diagrams, see the *Stateflow User's Guide*.

Stateflow Coder

Stateflow Coder provides the utilities to convert Stateflow blocks into code used by Real-Time Workshop.

Note Version 1.1 of xPC Target requires Stateflow version 4.0 on the R12 CD.

Stateflow Coder documentation - For information on state flow diagrams and the Stateflow Coder, see the *Stateflow User's Guide*.

DSP Blockset

The DSP Blockset provides the Simulink blocks to add digital signal processing functions to your Simulink model.

xPC Target is compatible with the DSP blockset.

Note Version 1.1 of xPC Target requires DSP Blockset version 4.0 on the R12 CD.

DSP Blockset documentation. For information on adding DSP blocks to your Simulink model, see the *DSP Blockset User's Guide*.

Dials & Gauges Blockset

The Dials and Gauges Blockset provides the Simulink blocks to create an instrument control panel. This instrument control panel acts as an interface to your real-time application.

Note Version 1.1 of xPC Target requires Dials & Gauges Blockset Version 1.1 on the Release 12 CD.

Dials & Gauges Blockset documentation - For information on creating a Dials & Gauges model, see the *Dials & Gauges Blockset User's Guide*.

Using This Guide

To help you effectively read and use this guide, this section provides a brief description of the chapters and a suggested reading path.

This section includes the following topics:

- “**Expected Background**”
- “**Organization**”

Expected Background

Users who read this book 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 xPC Target you do not need to program in C or other low-level programming languages to create, test, and deploy real-time systems.

If you are a new user - Begin with Chapter 1, “Introduction”. This chapter gives you an overview of the xPC Target features and xPC Target environment. Next, read and try the examples in Chapter 3, “Basic Procedures”.

If you are an experienced user - After you are familiar with using xPC Target, read or browse Chapter 6, “Environment Reference”, Chapter 7, “Target Object Reference”, and Chapter 8, “Scope Object Reference” for more detailed information about the commands in xPC Target. Next, read and try the examples in Chapter 4, “Advanced Procedures”.

Organization

The following table lists the organization of the xPC Target Getting Started Guide.

Chapter	Description
1, “Introduction”	Overview of the functions and features of xPC Target.
2, “Installation and Configuration”	Procedures to install xPC Target on the host computer, and configure the host computer for communication with the target computer.
3, “Basic Procedures”	Procedures for learning how to use xPC Target. The procedures are designed to help you become familiar with using xPC Target.
4, “Advanced Procedures”	Procedures for learning additional features in xPC Target.
5, “xPC Target Embedded Option”	Description and procedures for using the embedded option for stand-alone applications, and loading target applications from a device other than a 3.5 inch disk drive.
6, “Environment Reference”	Reference for the environment commands for setting up communication between the host and target computers.
7, “Target Object Reference”	Reference for the target object methods and properties.
8, “Scope Object Reference”	Reference for the scope object methods and properties

Conventions

To help you effectively read this guide, we use some conventions. Conventions are the ways of consistently formatting the text and graphics, and the meaning we use for common terms.

This section includes the following topics:

- “Terminology”
- “Typographical”

Terminology

Some technical terms have different meanings. The following table lists some of the terms used with real-time systems and the meaning we use with xPC Target.

Term	Definition
application	See target application.
build process	Process of generating C code from your Simulink model, compiling, linking, and downloading the generated code to create a <i>target application</i> .
execution	Running the <i>target application</i> on the target PC in real-time.
executable code	See target application.
kernel	Main real-time software component running on the target PC that manages the downloaded <i>target application</i> .
model	Simulink/Stateflow model.
parameter tuning	Process of changing block parameters and downloading the new values to a <i>target application</i> while it is running.
sample rate	Rate the <i>target application</i> is stepped in samples/second. Reciprocal of the <i>sample time</i> .

Term	Definition
sample time	Interval, in seconds, between the execution of a <i>target application</i> step.
signal logging	Acquire and save signal data created during a real-time execution.
signal monitoring	Get the values of one or more signals without time information.
signal tracing	Acquire and display packages of signal data during real-time execution.
simulation	Running a simulation of the Simulink/Stateflow model on the host PC in nonreal-time.
target application	Executable code generated from a Simulink/Stateflow model which can be executed by the xPC Target kernel on the target PC.

Typographical

Typographical conventions are ways of formatting the text to indicate terms, objects, and dialog between the user and the computer. The following table lists the notational conventions we use in the xPC Target Getting Started Guide.

Item	Convention to Use	Example
Example code	Monospace font	To start the application, type start (tg)
Function names	Monospace font	The addscope function creates a new scope object.
MATLAB output	Monospace font	MATLAB displays the output avgTET= 0. 000011
Keys	Boldface with an initial capital letter	Press Return .
Menu names, menu items, and command buttons	Boldface with an initial capital letter	From the File menu, click Open . Click Update .
New terms	<i>Italics</i>	The <i>target application</i> is downloaded from the <i>host computer</i> .

Introduction

What Is xPC Target?	1-3
Features of xPC Target	1-4
Real-Time Kernel	1-4
Real-Time Application	1-6
Signal Acquisition and Analysis	1-6
Parameter Tuning	1-7
Hardware Environment	1-8
Host PC	1-8
Target PC	1-8
Host-Target Connection.	1-9
Input/Output Driver Support	1-11
Software Environment	1-12
Host-Target Communication	1-12
Rapid Prototyping Process	1-13
Embedded Process	1-15
User Interaction	1-18
xPC Target Graphical Interface	1-19
MATLAB Command Line Interface	1-20
Target PC Command Line Interface	1-21
Simulink External Mode Interface.	1-21
Simulink Dials & Gauges Interface	1-22
Web Browser Interface	1-22

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

This chapter includes the following sections:

- **“What Is xPC Target?”** - Host-target PC solution for prototyping, testing, and deploying real-time systems
- **“Features of xPC Target”** - Real-time kernel, real-time application, signal acquisition and analysis, and parameter tuning
- **“Software Environment”** - Rapid prototyping process and embedded process
- **“Hardware Environment”** - Host PC and target PC
- **“User Interaction”** - Graphical, command line, Web browser, Simulink external mode, Dials & Gauges

What Is xPC Target?

xPC Target is a host-target PC solution for prototyping, testing, and deploying real-time systems. It is an environment where the host and target computers are different computers.

In this environment you use your desktop PC as a host computer with MATLAB[®], Simulink[®], and Stateflow[®] (optional) to create models using Simulink blocks and Stateflow diagrams. After creating a model, you can run simulations in nonreal-time.

You can then use your host computer with Real-Time Workshop[®], Stateflow Coder (optional) and a C compiler to create executable code. After creating the executable code, you can run your target application in real time on a second PC compatible system.

- **Special hardware requirements** - The xPC Target software requires a host PC, target PC, and for I/O, the target PC must also have I/O boards supported by xPC Target.
- **Special software requirements** - The xPC Target software requires either a Microsoft Visual C/C++ compiler (version 5.0 or 6.0) or a Watcom C/C++ compiler (version 10.6 or 11.0). In addition, xPC Target requires, MATLAB, Simulink, and Real-Time Workshop.
- **xPC Target Embedded Option requirements** - The xPC Target Embedded Option is a separate product that requires an additional licence from The MathWorks. With this additional licence, you can deploy an unlimited number of real-time applications for stand alone operation.

This option allows you to boot the target PC from an alternate device other than a floppy disk drive such as a hard disk drive or flash memory. It also allows you to create stand-alone applications on the target PC independent from the host PC.

Features of xPC Target

The xPC Target software environment includes many features to help you prototype, test and deploy real-time systems.

This section includes the following topics:

- **“Real-Time Kernel”** - BIOS, BIOS-extension, kernel, and loader
- **“Real-Time Application”** - Memory model, and task execution time
- **“Signal Acquisition and Analysis”** - Signal monitoring, logging to the MATLAB workspace, and signal tracing on the host PC or target PC screen
- **“Parameter Tuning”** - Interactive, scripts and batch procedures

Real-Time Kernel

xPC Target does not require DOS, Windows, Linux, or any another operating system on the target PC. Instead, you boot the target PC with a special boot disk that includes the highly optimized xPC Target kernel.

Target boot disk - The boot disk eliminates the need to install software, modify existing software configurations, or access the hard disk on the target PC. This arrangement allows you to use the target PC for testing real-time applications, and then when you have finished your tests, you can use the target PC again as a desktop computer. Software is not permanently installed on the target PC unless you are deliberately using the xPC Target Embedded Option and install a stand-alone application on the hard disk or flash memory.

Target PC BIOS - Selecting a newer BIOS allows you to customize settings for better control over the real-time behavior of the system. For example, turn off the external and CPU-cache, suppress the check for a keyboard, and switch off any power save features.

The xPC Target kernel runs only on a PC compatible system and a key component of every PC compatible system is the BIOS. The BIOS is the only software component which is needed by the xPC Target kernel.

After the BIOS is loaded, it searches the target boot disk for a bootable image (executable). This bootable image includes a 16-bit part and a 32-bit part. The 16-bit part runs first because the CPU is still in real mode. It prepares the descriptor tables and in addition to other things switches the CPU to protected mode. Next, the 32-bit part runs. It prepares the target PC environment for running the kernel and finally starts the kernel.

After loading the kernel, the target PC does not make calls to the BIOS or DOS functions. The resources (for example, interrupt controller, UART, and counters) on the CPU motherboard are addressed entirely through I/O addresses.

Real-Time Kernel - After the kernel starts running, it displays a welcome message with information about the host-target connection. The kernel activates the application loader and waits to download a target application from the host PC. The loader receives the code, copies the different code sections to their designated addresses, and sets the target application ready to start. You can now use xPC Target functions and other utilities to communicate with the target application.

It is important to note, that after the CPU switches to protected mode (32-bit), none of the xPC Target components switch the CPU back to real mode (16-bit).

The generated real-time application and the real-time kernel are compiled as Windows NT applications with a flat memory model. This provides full 32-bit power without time consuming 16-bit segment switching and DOS extenders.

Target PC heap - The initialization code of the target application reserves the remaining unused RAM as heap. The memory available for the heap is displayed on the left side of the target screen as Memory. By default, it is 4 MB less than the entire RAM installed (1 MB for the application; 3 MB for the kernel). Normally, the largest part of the heap is used by signal logging because logging acquires and stores data during the entire run.

You can define the amount of memory available for data logging in the Simulation Parameters dialog box. See “Entering the Simulation Parameters” on page 3-8.

Real-Time Application

A real-time application (target application) is created from a Simulink/Stateflow model with Real-Time Workshop, Stateflow Coder, and xPC Target. Applications created with Real-Time Workshop and xPC Target run in real-time on a standard PC without using a standard operating system.

The target application runs in real time on the target PC and has the following characteristics:

- **Memory model** - The target application is compiled as a Windows NT application with the flat memory model. This provides full 32-bit power without time consuming 16-bit segment switching and DOS extenders. Also, it does not rely on DOS or any other Microsoft operating system.
- **Task Execution time** - The target application is capable of high-speed, real-time task execution. A small block diagram can run with a sample time as fast as 10 μ s (100 MHz). Model size, complexity, and target PC hardware affect maximum speed (minimal sample time) of execution.

For more information on creating a target application, see “Creating the Target Application” on page 3-7.

Signal Acquisition and Analysis

Signal acquisition is through the real-time kernel. Signal data from your real-time application is stored in RAM on the target PC, and can then be used for analysis. xPC Target supports the following types of signal acquisition:

- **Signal Monitoring** - This is the process for acquiring signal data without time information. In this mode, you can get the current value of one or more signals. The data is not acquired in the real-time task but in the background task. The advantage of this process is that collecting data does not add any computational load to running the real-time application.

For example, if you have a LED Gauge in a Simulink model on the host PC, you could use signal monitoring to display the status of the signal.

- **Signal Logging** - This is the process for acquiring signal data during a real-time run. The data is collected in the real-time task and acquired samples are associated with a time stamp. After the run reaches its final time or you manually stop the run, the host PC makes a request to upload data from the target PC. You can then visualize signals by plotting data on the host PC, or you can save data to a disk.
- **Signal Tracing** - This is the process of acquiring and visualizing signals during a real-time run. The data is collected in the real-time task and acquired samples are associated with a time stamp. It allows you to acquire signal data and visualize it on the target PC or to upload the signal data and visualize it on the host PC while the target application is running. The flexibility of this acquisition type is very similar to the behavior of a digital oscilloscope.

For information on the various ways to acquire signal data with xPC Target, see “User Interaction” on page 1-18.

Parameter Tuning

Most Simulink blocks have parameters that you can change before or while your target application is running. For example, parameters include the amplitude and frequency of a sine wave.

- **Interactive** - xPC Target supports interactive tuning of parameters while the target application is running in real time. The changes to parameters are immediately reflected in the signal outputs.
- **Scripts and batch procedures** - xPC Target also includes commands to change parameters during a run or between runs. By writing a script that incrementally changes a parameter and monitors a signal output, you can optimize the value of that parameter.

For information on the various ways to tune parameters with xPC Target, see “User Interaction” on page 1-18.

Hardware Environment

The hardware environment consists of a host computer, target computer, I/O boards on the target computer, and a serial or network connection between the host and target computers.

This section includes the following topics:

- **“Host PC”** - Desktop PC, or notebook PC
- **“Target PC”** - Desktop PC, industrial PC, PC 104, or CompactPCI
- **“Host-Target Connection”**- RS232 serial or TCP/IP network
- **“Input/Output Driver Support”** - Analog, digital, CAN, GPIB, RS232, counters, timers, and signal conditioning

Host PC

You can use any PC that runs Windows 95, Windows 98, Windows 2000, or Windows NT on the host PC. Also, it must contain a 3.5-inch floppy disk drive, and a free serial port or an Ethernet adapter card.

The host PC can be one of the following:

- **Desktop PC**
- **Notebook PC**

For more details on the requirements of the host PC, see “Host PC” on page 2-3.

Target PC

You can use virtually any PC with an Intel 386/486/Pentium or AMD K5/K6/Athlon processor as the target computer with a floating-point unit (FPU) present. Also, it must contain a 3.5 inch floppy disk drive, and a free serial port or an Ethernet adapter card. Using the xPC Target Embedded Option, you can transfer files from the 3.5 inch disk to a hard disk or flash memory.

The target PC can be one of the following:

- **Desktop PC** - This computer is booted from a special target boot disk created by xPC Target.

When you boot the target PC from the target boot disk, xPC Target uses the resources on the target PC (CPU, RAM, and serial port or network adapter) without changing the files already stored on the hard drive.

After you are done using your desktop computer as a target PC, you can easily reboot your computer without the target boot disk. You can then resume normal use of your desktop computer using the pre-existing operating system and applications.

- **Industrial PC** - This computer is booted from a special target boot disk, or with the xPC Target Embedded Option, booted from a hard disk or a flash memory.

When using an industrial target PC, you can select PC104, PC104+, CompactPCI, or single-board computer (SBC) hardware.

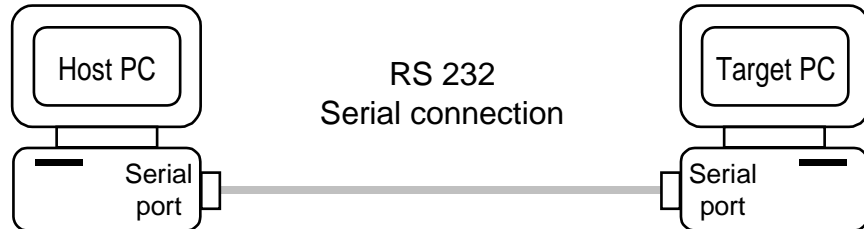
You do not need any special target hardware. However, the target PC must be a fully compatible system and contain a serial port or Ethernet card. For more details on the requirements of the target PC, see “Target PC” on page 2-4.

Host-Target Connection

xPC Target supports two connection-and-communication protocols between the host PC and the target PC: serial and network.

Serial - The host and target computers are connected directly together with a serial cable using their RS232 ports. This cable is wired as a null modem link that can be up to 5 meters long. We provide a null modem cable with the xPC Target software.

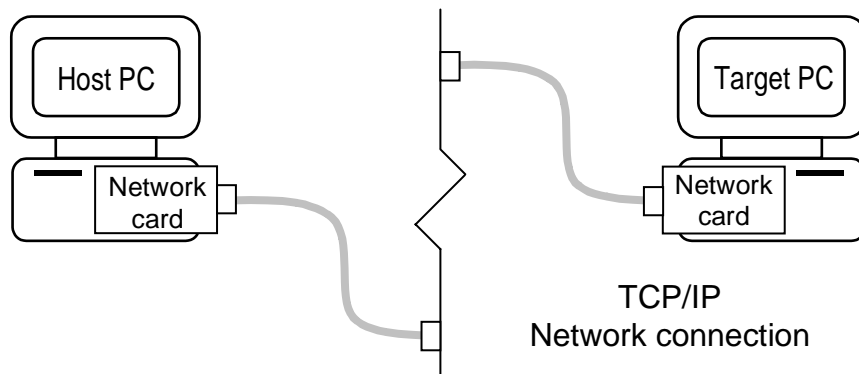
The transfer rate is variable between 1200 and 115200 Baud.



For detailed information to setup the hardware and software for serial communication, see “Serial Communication” on page 2-12.

Network - The host and target computers are connected through a network. The network can be a LAN, the Internet, or a direct connection using a cross-over Ethernet cable. Both the host and target computers are connected to the network with Ethernet adapter cards. xPC Target uses the TCP/IP protocol for communication.

When using a network connection, the target PC can use the Ethernet adapter card provided with xPC Target or one of the supported cards. The data transfer rate is limited to 10 Mbit/s. For a list of supported cards, see “Hardware for Network Communication” on page 2-16.



For detailed information to setup the hardware and software for network communication, see “Network Communication” on page 2-15.

Input/Output Driver Support

xPC Target supports a wide range of I/O boards. The list of supported I/O boards includes ISA, PCI, PC/104, and CompactPCI hardware. The drivers are represented by Simulink blocks. Your interaction with the drivers is through these Simulink blocks and the parameter dialog boxes.

I/O board library - The I/O driver library contains Simulink blocks for xPC Target. You drag-and-drop a block from the I/O library and connect I/O drivers to your model the same way as you would connect any standard Simulink block.

Input/Output support - The I/O device library supports over 40 standard boards. Input/Output boards plug into the target PC expansion bus, or are modules in PC104 and industrial PC computers. xPC Target supports the following I/O functions:

- **Analog input (A/D) and analog output (D/A)** - Interface sensors and actuators to your target application
- **Digital input and output** - Interface to switches, on/off devices, and communicate information in parallel
- **RS232 support** - COM1 or COM2 ports for serial communication
- **CAN support** - CAN-AC2, CAN-AC2-PCI, and CAN-AC2-104 boards from Softing GmbH

The xPC Target CAN drivers enable you to interface with a CAN fieldbus network to provide communication through a CAN network between real-time applications and remote sensors and actuators. The xPC Target CAN drivers are compatible with CAN specification 2.0A and 2.0B and use the dynamic object mode.

- **GPIO support** - Special RS232 drivers support communication with a GPIO control module from National Instruments
- **Counter** - Pulse and frequency code modulation applications
- **Watchdog** - Monitor an interrupt or memory location, and reset the computer if an application does not respond
- **Incremental encoder** - Change motion into numerical information for determining position, direction of rotation, and velocity
- **Shared memory**. Use with multiprocessing applications

Software Environment

The software environment is a place to design, build, and test a target application in nonreal-time and real-time. It also includes communication between the host and target computers.

This section includes the following topics:

- **“Host-Target Communication”** - Control the target application, upload signal data, and download parameter values.
- **“Rapid Prototyping Process”** - Design and build a target application on a host PC. Download the target application to a target PC. Run and test the target application with interactive tuning of parameters and acquisition of data.
- **“Embedded Process”** - Boot the kernel from a device other than the floppy disk drive, but download the target application from the host PC. Or boot the kernel/application from the floppy disk drive or another device, and run the target application completely separate from the host PC.

Host-Target Communication

Whether using a serial connection (RS232) or using a network connection (TCP/IP), information is exchanged between the host PC and target PC. This information includes:

- **Target application.** - Download a target application from the host to the target computer.
- **Control** -Change properties and control the target application. This includes starting and stopping the target application, changing sample and stop times, and getting information about the performance of the application and CPU.
- **Signal data** - Upload signal data from the host computer for analysis after the target application is finished running, or view signal data during the run.
- **Parameter values** - Download parameter values to the target computer between runs or during a run.

Rapid Prototyping Process

Design and build a target application on a host PC, and then run and test the target application on a target PC. xPC Target functions include interactive control of the target application, acquisition of signal data, and tuning of parameters while running in real time.

The rapid prototyping process includes the following sequence of tasks:

- 1 **Create a Simulink/Stateflow model** - You create block diagrams in Simulink using simple drag-and-drop operations, then you enter values for the block parameters and select sample rates. If you use continuous-time components, you also need to select an integration algorithm.
- 2 **Simulate the model in nonreal-time** - Simulink uses a computed time vector to step the 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.

- 3 **Create an executable target application** - Real-Time Workshop, Stateflow Coder, xPC Target, and a C compiler create the target application that runs on the target PC. This real-time application uses the initial parameters from the Simulink model that were available at the time of code generation.
- 4 **Execute the target application in real-time** - The target PC is booted using a special boot disk that loads the xPC Target real-time kernel. After booting the target PC, you can build and download a real-time application.

xPC Target provides the necessary software that makes use of real-time resources on the target PC hardware. Based on your selected sample rate, xPC Target uses interrupts to step the model at the proper rate. With each new interrupt, the target application computes all of the block outputs from your model.

5 Acquire signals and tune parameters - You acquire signal data using xPC Target scopes and Gauge blocks. Scopes are created on the target PC by:

- Adding xPC Target Scope blocks to your Simulink model
- Using the xPC Target Scope manager window
- Commands in the MATLAB window
- Commands in the target PC command window
- Adding Gauge blocks to a second Simulink model
- Using the xPC Target Web browser interface

Note xPC Target does not support normal Simulink scope blocks in external mode. Instead, use xPC Target scope blocks.

You can tune parameters using:

- Commands in the MATLAB window
- Commands in the target PC command window
- Simulink in external mode
- Adding Dial blocks to a second Simulink model
- Using the xPC Target Web browser interface.

Embedded Process

Often, control system and digital signal processing applications are developed for use in production where a limited number of deployed systems are required. Whether deploying one or one hundred systems, the xPC Target Embedded Option provides a convenient approach that allows you to implement your system on low cost PC hardware.

When you have completed development and testing, you can use the target application as a real-time system that runs on a dedicated target PC without the need to connect to the host computer.

The xPC Target Embedded Option consists of two modes of operation. In each case, the target PC boots into DOS, starts the DOS program xpcboot.com from autoexec.bat, and then starts the kernel from xpcboot.com.

- **“DOSLoader Mode”** - Run the kernel from a device other than the floppy disk drive such as a hard disk or flash disk, and download the target application from the host PC.
- **“(DOSLoader)/Standalone Mode”** - Run both the kernel and the target application from the floppy disk drive on the target PC, or optionally boot from a device other than the floppy disk drive. The target application runs completely independent from the host PC.

Note The xPC Target Embedded Option is a separate product that requires an additional licence from The MathWorks. With this additional licence you can deploy an unlimited number of real-time applications for stand alone operation.

For more information on the xPC Target Embedded Option, see “xPC Target Embedded Option” on page 5-1.

DOSLoader Mode

The **DOSLoader** mode allows you to boot your target computer from devices other than a 3.5-inch floppy disk drive. For example, the boot device can be a hard disk drive or flash memory. The target application is still downloaded from the host PC. You can also use this mode to boot from a floppy disk drive, but there is no advantage to use this method over using a normal target boot disk.

- 1 Select the DOSLoader mode from the xPC Target Setup window.
- 2 Create a new target boot disk.
- 3 Copy DOS system files and utilities to the target boot disk. In the `autoexec.bat` file, remove the line that loads `xpcboot.com`.
- 4 Boot the target PC with the target boot disk from the floppy disk drive.
- 5 Copy files to the alternate boot device. In the `autoexec.bat` file, replace the line that loads `xpcboot.com`.
- 6 Remove the target boot disk and reboot the target PC from the alternate boot device.
- 7 Build a target application and download it to the target PC.

For more information on the xPC Target Embedded Option, see “xPC Target Embedded Option” on page 5-1

(DOSLoader)/Standalone Mode

The **Standalone** mode combines the target application with the kernel and boots them together on the target PC from a 3.5 inch floppy disk drive, hard disk drive, or flash memory. The host PC does not have to be connected to the target PC.

- 1 Select the StandAlone mode from the xPC Target Setup window.
- 2 Build a kernel/target application.
- 3 Copy DOS system files, utilities, kernel/application files and helper files to the boot disk.

Note If booting from an alternate boot device, in the autoexec. bat file, remove the line that loads xpcboot. com.

- 4 Boot the target PC with the target boot disk from the floppy disk drive.

Note If booting from an alternate boot device, copy files to the alternate boot device. In the autoexec. bat file, replace the line that loads xpcboot. com. Remove the target boot disk and reboot the target PC from the alternate boot device.

For more information on the xPC Target Embedded Option, see “xPC Target Embedded Option” on page 5-1

User Interaction

The xPC Target environment has an intuitive and modifiable interface. It uses an object-oriented structure with properties and methods. Because of this open structure, there are several ways to interact with your target application.

This section includes the following topics:

- **“xPC Target Graphical Interface”** - Use the xPC Target GUIs to set environment properties and create xPC Target scopes
- **“MATLAB Command Line Interface”** - Enter xPC Target functions on the host PC.
- **“Target PC Command Line Interface”** - Enter xPC Target functions on the target PC
- **“Simulink External Mode Interface”** - Connect a Simulink block diagram to the target application for parameter tuning
- **“Simulink Dials and Gauges Interface”** - Create a second Simulink model with Dials, Gauges, and special xPC Target interface blocks
- **“Web Browser Interface”** - Use Microsoft Internet Explorer or Netscape Navigator to connect to the target computer from any computer on the network

The following table compares the different interfaces supported by xPC Target.

Interface	Environment properties	Control	Signal Acquisition	Parameter Tuning
xPC Target GUI	X		X	
MATLAB Command Line	X	X	X	X
Target PC Command Line		X	X	X
Simulink External Mode		X		X
Web Interface		X	X	X
Dials & Gauges			X	X

xPC Target Graphical Interface

xPC Target offers graphical user interfaces (GUIs) for setting up the xPC Target environment, and for tracing signals while running real-time applications. These GUIs are built using xPC Target commands and MATLAB Handle Graphics®.

There is no xPC Target GUI for controlling the target application or tuning parameters. Use one of the other methods of interaction for these functions.

The xPC Target graphical interface includes the following functions:

- **Environment** - Use the xPC Target Setup window to change properties in the xPC Target environment. Properties include, communication between the host and target computers, and entering the type and location of your C compiler.

For more information on environment properties, see “Serial Communication” on page 2-12, “Network Communication” on page 2-15, and the “Environment Reference” on page 6-1.

- **Signal acquisition** - Use the Scope Manager window to interactively add scopes of type host or target, add or remove signals, and set triggering modes. A scope created on the target PC acquires data from the target application and stores the data for display.

A scope with type host continuously uploads the signal traces and displays them on the host PC.

A scope with type target displays the signal traces on the target PC. Because xPC Target uses highly optimized graphic routines, signal tracing has the same fast display and update rates that you normally observe when using digital oscilloscopes.

An alternative to using the Scope Manager window is to add special xPC Target scope blocks to your Simulink model. These blocks create scopes on the target PC during initialization of the target application after the download process.

For more information on using scopes, see “Signal Tracing with xPC Target GUI” on page 3-26 and “Signal Tracing with xPC Target GUI (Target Manager)” on page 3-31.

MATLAB Command Line Interface

You can interact with the xPC Target environment through the MATLAB command line interface. Enter xPC Target commands in the MATLAB window on the host PC. You can also write your own M-file scripts that use xPC Target functions for batch testing procedures.

xPC Target has more than 40 MATLAB functions for controlling the target application from the host computer. These functions define, at its most basic level, what you can do with the xPC Target system.

Some tasks are difficult to implement with graphical interface objects. The GUIs that we provide with xPC Target are for the most common tasks. They use the functions but do not extend their functionality. The command line interface provides an interactive environment that you can extend.

The MATLAB command line interface includes the following functions:

- **Environment** - Directly change the environment properties without using a GUI.
- **Control** - Download, start, and stop real-time applications. Change sample times without regenerating code. Get statistical performance information during or after the last run.
- **Signal acquisition** - Trace signals for viewing while the real-time application is running. Transfer signal data to the MATLAB workspace by uploading signal data between runs.
- **Parameter tuning** - Change parameters while the real-time application is running. Also, use xPC Target commands to change parameters in between runs or during a run. This allows you to batch process many runs at one time.

For a complete list of MATLAB functions to use with xPC Target, refer to “Environment Reference” on page 6-1, “Target Object Reference” on page 7-1, and “Scope Object Reference” on page 8-1

Target PC Command Line Interface

You can interact with the xPC Target environment through the target PC command window. Enter commands in the command line on the target PC. This interface is useful with stand-alone applications that are not connected to the host PC.

The target PC command line interface includes the following functions:

- **Control** - Start and stop the target application, change the stop time and sample time
- **Signal acquisition** - Acquiring signal data is limited to viewing signal traces and signal monitoring on the target PC screen.
- **Parameter tuning** - Changing parameters is limited to changing the scalar parameters in your model.

Simulink External Mode Interface

Use Simulink in external mode to connect your Simulink block diagram to your target application. The block diagram becomes a graphical interface to the target application running in real time. By changing parameters in the Simulink blocks you also change parameters in the target application.

The Simulink external mode interface includes the following functions:

- **Control** - Limited to connecting the Simulink block diagram, starting and stopping the target application
- **Parameter tuning** - Select external mode, and change parameters in the target application by changing parameters in the Simulink parameter dialog boxes. Once you change a value, the new value is immediately downloaded to the target PC and replaces the existing parameter while the target application continues to run.

Note xPC Target does not support data acquisition through Simulink External Mode. Instead, use xPC Target scope blocks.

What Is External Simulation Mode? External simulation mode is a feature of the Real-Time Workshop (RTW). It offers an easy way to change parameters in a target application regardless of whether a target application is running or not:

- If a target application is not running, it allows you to prepare a model with a new set of parameters before the next run.
- If a target application is running, it allows you to change parameters and immediately see what effect changing parameters has on the behavior of your generated code.

Simulink Dials and Gauges Interface

Use Dials, Gauges, and special xPC Target interface blocks to create a second Simulink model. This second Simulink model runs in normal mode and in nonreal-time. The special xPC Target interface blocks connect the second Simulink model to the target application, which runs in real time.

The Simulink Dials and Gauges interface includes the following functions:

- **Signal acquisition** - Use Gauge blocks to visualize and animate signal data.
- **Parameter tuning** - Use Dial blocks to change parameters in your model.

Web Browser Interface

If the target PC is connected to a network (TCP/IP), you can use a Web browser to interact with the target application from any computer connected to a network.

The Web browser interface includes the following functions:

- **Control** - Start and stop the target application, change the stop time and sample time
- **Signal acquisition** - Signal tracing is limited to viewing a snapshot of a scope screen captured from the target PC screen. Add scopes of type target, add or remove signals, and set triggering modes. Also, you can monitor signal values.
- **Parameter tuning** - Change parameters in an HTML form, and then submit that form to make the changes in your target application.

Installation and Configuration

System Requirements	2-3
Host PC	2-3
Target PC	2-4
 Installation on the Host PC	 2-7
Getting or Updating Your License	2-7
CD-ROM Installation	2-8
Web Download Installation	2-8
Files on the Host PC Computer	2-10
 Initial Working Directory	 2-11
Setting Your Working Directory from the Desktop Icon	2-11
Setting Your Working Directory from Within MATLAB	2-11
 Serial Communication	 2-12
Hardware for Serial Communication	2-12
Environment Properties for Serial Communication	2-13
 Network Communication	 2-15
Hardware for Network Communication	2-16
Ethernet Card for a PCI-Bus	2-18
Ethernet Card for an ISA-Bus	2-18
Environment Properties for Network Communication	2-20
 Target Boot Disk	 2-23
Current Properties on the Target Boot Disk	2-25
 Testing and Troubleshooting the Installation.	 2-26
Testing the Installation	2-26
If You Still Need More Help	2-31

The software environment for xPC target uses two separate computers. Because of this added complexity, installation and configuration are more involved.

This chapter includes the following sections:

- **“System Requirements”** - Select a host PC and target PC
- **“Installation on the Host PC”** - Get a valid license for xPC Target, and a separate license for the xPC Target Embedded Option. Install from a CD or download from the Web
- **“Serial Communication”** - Select for an easy and inexpensive installation
- **“Network Communication”** - Select for faster data transfer rates and longer connections
- **“Target Boot Disk”** - Boot the kernel on the target PC and establish a connection with the host PC
- **“Testing and Troubleshooting the Installation”** - Test the installation

System Requirements

The hardware and software requirements are different for the host and target computers.

This section includes the following topics:

- **“Host PC”** - Desktop or notebook PC
- **“Target PC”** - Desktop, industrial PC, PC/104, and Compact PC

Host PC

The host PC is usually your desktop computer where you install MATLAB, Simulink, Stateflow, Stateflow Coder, Real-Time Workshop, and xPC Target. Also, you can use a notebook computer without slots as the host computer.

The following table lists the minimum software xPC Target requires on your host PC system. For a list of optional software products related to xPC Target, see “Related Products” on page -xv.

Table 2-1: Software Requirements for the Host PC

Software	Description
Operating system	Windows 95, Windows 98, Windows 2000 or Windows NT 4.0.
MATLAB	Version 6.0
Simulink	Version 4.0
Real-Time Workshop	Version 4.0
C language compiler	Microsoft Visual C/C++ versions 5.0 or 6.0. Watcom C/C++ versions 10.6 or 11.0.
xPC Target	Version 1.1

The following table lists the minimum resources xPC Target requires on the host PC system.

Table 2-2: Hardware Requirements for the Host PC

Hardware	Description
Communication	One free serial port (COM1 or COM2) with a 9-pin or 25-pin D-sub connector, or an Ethernet card connected to a network.
CPU	Pentium, Athlon or higher.
Peripherals	Hard disk drive with 60 Mbytes of free space. One 3.5-inch floppy disk drive. CD-ROM drive.
RAM	32 Mbytes or more.

Target PC

The target PC has to be a PC compatible system. In many cases you can use a second desktop computer as the target PC, but also you can use an industrial system like a PC/104 or CompactPCI as the target computer.

The following table lists the software xPC Target requires on the target PC system.

Table 2-3: Software Requirements for the Target PC

Software	Description
Operating system	None. If you have an operating system installed, it is not affected.
BIOS	PC compatible.

The following table lists the minimum hardware resources xPC Target requires on the target PC system.

Table 2-4: Hardware Requirements for the Target PC

Hardware	Description
Chip set	PC compatible with UART (For example, 16550), programmable interrupt controller (For example, 8259), keyboard controller, and counter (For example, 8254).
Communication	One free serial port (COM1 or COM2) with a 9-pin or 25-pin D-sub connector, or an Ethernet card connected to a network. An Ethernet card is provided with xPC Target.
CPU	Intel 386 with a floating-point processor, Intel 486/ Pentium or AMD K5/K6. We recommend a Pentium, K6 or higher CPU. xPC Target does not support DEC Alpha computers.
Keyboard and mouse	<p>Needed to control the target PC when you create stand-alone applications.</p> <p>Note If a keyboard is not connected, the BIOS may display an error message (keyboard failure). With a newer BIOS, you can use the BIOS setup to skip the keyboard test.</p>
Monitor	We recommend using a monitor, but it is not necessary. You can get all of the target information using xPC Target functions on the host PC.
Peripherals	<p>One 3.5 inch floppy disk drive. A hard disk drive is not required.</p> <p>Note If you install the xPC Target Embedded Option, you can copy files to a hard disk or flash memory and boot from that device.</p>
RAM	8 Mbytes or more.

Random Access Memory (RAM) - xPC Target works with PC compatible computers that use inexpensive dynamic RAM, unlike many nonPC compatible target computers that use expensive static RAM. You can acquire several megabytes of data during a run depending on how much memory you install in the target PC.

PC compatible target computers - xPC Target supports the following PC-compatible hardware (form factors):

- PC ISA
- PC PCI
- PC/104 and PC/104+
- CompactPCI

I/O boards - You can install inexpensive I/O boards in the PCI or ISA slots of the target PC. These boards provide a direct interface to the sensors, actuators, or other devices for real-time control or signal processing applications.

For a list of I/O functions supported by xPC Target, see “Input/Output Driver Support” on page 1-11.

Installation on the Host PC

You install the xPC Target software entirely on the host PC. Installing software on the target PC is not necessary. We distribute the xPC Target software on a CD-ROM or as a file you download from the Web.

This section includes the following topics:

- **Getting or Updating Your License**
- **CD-ROM Installation**
- **Web Download Installation**

The xPC Target family of software includes options that you can purchase and add later to the xPC Target environment.

xPC Target Embedded Option - With the xPC Target Embedded Option active, you have additional choices for the type of target boot disk. You can choose from **BootFloppy**, **DosLoader**, and **StandAlone**. See the “Embedded Process” on page 1-15.

Getting or Updating Your License

Before you install xPC Target or the xPC Target Embedded Option, you must have a valid License File or Personal License Password (PLP). The License File or Personal License Password identifies the products you purchased from The MathWorks and you are permitted to install and use.

When you purchase a product, The MathWorks sends you a License File or Personal License Password (PLP) in an e-mail message. If you have not received a PLP number, contact the MathWorks.

Internet	http://www.mathworks.com/mla Log into MATLAB Access using your last name and Access number. Follow the license links to determine your PLP number
E-mail	mailto:service@mathworks.com. Include your license number
Telephone	508-647-7000. Ask for Customer Service
Fax	508-647-7001. Include your license number

CD-ROM Installation

We distribute xPC Target version 1.1 on The MathWorks R12.0 CD with the general installation program.

After you get a valid Personal License Password (PLP), you can install the xPC Target software. For detailed information about the installation process, see the MATLAB *Install Guide for PC*.

- 1 Insert the R12.0 CD-ROM into the host CD drive.

After a few seconds, the installation program starts automatically. If the installation program does not start automatically, run `setup.exe` on the CD-ROM.

- 2 Follow the instructions on each dialog box.

The xPC Target installation is now complete.

Your next task is to set up the xPC Target environment for either serial or network communication. See “Serial Communication” on page 2-12 or “Network Communication” on page 2-15.

Web Download Installation

We distribute xPC Target as a single, self-extracting file. After you get a valid Personal License Password (PLP), you can install the xPC Target software. See “Getting or Updating Your License” on page 2-7.

To download xPC Target from the Internet, and install it on the host computer, use the following procedure:

- 1 In the Web browser window, enter the following address

`http://www.mathworks.com`

- 2 On the right side of the page, click the link labeled **Downloads**. On the Downloads Web page, click the link labeled **download products**.

The MATLAB Access Web page opens.

- 3 Enter your last name and your MATLAB Access number. Click the **Login** button.

The Downloads Web page opens.

- 4 From the left list, select the **PC Windows** check box, and then click the **Continue** button. From the Select Your Products list, select the **xPC Target** check box, and then click the **Continue** button.
- 5 On the next Web page, click the **xPC Target** link. In the File Download dialog box, select **Save this file to disk**, and select the directory where you installed MATLAB.

Your browser downloads the file `xPC_Target.exe` to your computer.

- 6 Double-click the self-extracting file `xPC_Target.exe`.

The install program copies extracted files to a temporary directory and starts the MATLAB installation program.

- 7 Follow the instructions on each dialog box.

After MATLAB finishes the installation, the install program deletes all of the files from the temporary directory.

The xPC Target installation is now complete.

Your next task is to set up the xPC Target environment for either serial or network communication. See “Serial Communication” on page 2-12 or “Network Communication” on page 2-15.

Files on the Host PC Computer

When using xPC Target, you may find it helpful to know where files are located:

- **MATLAB working directory** - Simulink models (model . mdl), xPC Target applications (model . dl m)
Note select a working directory outside of the MATLAB root. See “Initial Working Directory” on page 2-11
- **RTW working directory** - The RTW C-code files (model . c, model . h) are in a subdirectory called model _xpc_rtw.

xPC Target uses the directories and files located in

C: \MATLABR00T\Tool box\rtw\targets\xpc:

- **target** - Files and functions related to the xPC Target kernel and build process
- **xpc** - Functions related overall to xPC Target, methods for target objects, and methods for scope objects
- **xpcdemos** - Simulink models and M-file demos.

Initial Working Directory

You should set your MATLAB working directory outside of the MATLAB root directory. The default MATLAB root directory is `c:\matlab`.

If your MATLAB working directory is below or inside the MATLAB root, files created by Simulink, Real-Time Workshop, and Real-Time Windows Target are mixed with the MATLAB directories. This mixing of files could cause you file management problems when deleting unwanted files.

Setting Your 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 initially use. Make sure you choose a directory outside of the MATLAB root directory.
- 3 Click **OK**, and then start MATLAB. To check your working directory, type
`pwd` or `cd`

Setting Your Working Directory from Within MATLAB

An alternative, but temporary, procedure for setting your MATLAB working directory is:

- 1 In the MATLAB command window, type
`cd c:\mwd`
- 2 To check your working directory, type
`pwd` or `cd`

Serial Communication

Before you can create and run a target application, you need to set up the connection between your host and target computers. You can use either serial or network communication.

This section includes the following topics:

- **“Hardware for Serial Communication”**
- **“Environment Properties for Serial Communication”**

For using network communication, see “Network Communication” on page 2-15.

Advantages of Serial Communication - A host-to-target connection using serial RS232 communication has advantages over network TCP/IP communication: inexpensive, easy to install, and always available.

Hardware for Serial Communication

Before you install the xPC Target software and configure it for serial communication, you must install the following hardware:

- **Null modem cable** - You connect the host and target computers with the null modem cable supplied by The MathWorks with the xPC Target software. You can use either the COM1 or COM2 ports.
- **I/O boards** - If you use I/O boards on the target PC, you need to correctly install the boards. See the manufactures literature for installation instructions.

Environment Properties for Serial Communication

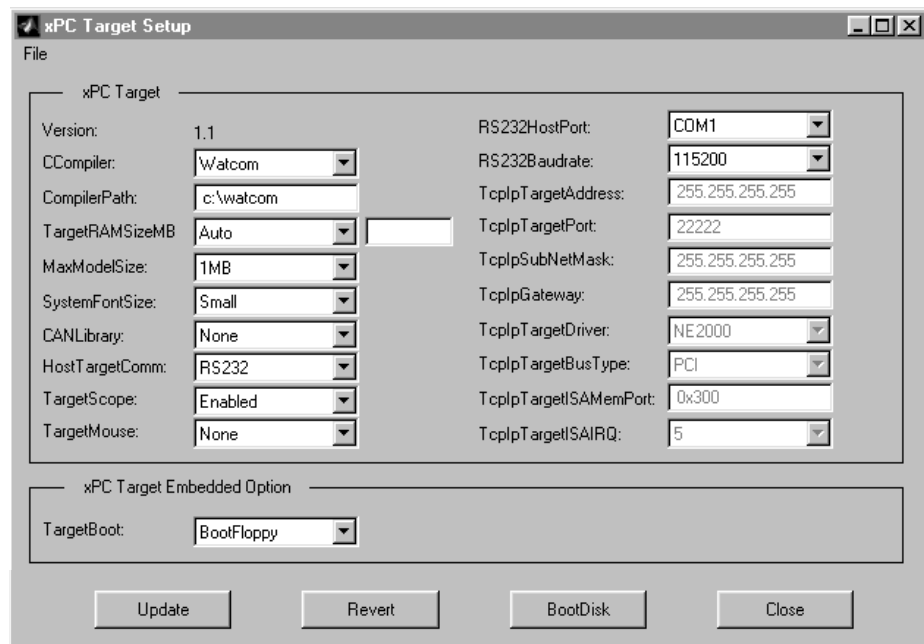
The xPC Target environment is defined by a group of properties. These properties give xPC Target information about the software and hardware products that it works with. You might change some of these properties often, while others you would change only rarely.

After you have installed xPC Target, you can set the environment properties for the host and target computers. You need to change these properties before you can build and download a target application.

To change the environment properties using a graphical user interface, use the following procedure.

- 1 In the MATLAB window, type
`xpcsetup`

The xPC Target Setup window opens.



The xPC Target Setup window has two sections:

- xPC Target
- xPC Target Embedded Option

If your license does not include the embedded option, the **TargetBoot** list is disabled (grayed-out) with **Boot Floppy** as your only choice. With the xPC Target Embedded Option installed, you have the additional choices of **DOSLoader** and **StandAlone**.

- 2 From the **CCompiler** list, choose either **VisualC** or **Watcom**.
- 3 In the **CompilerPath** box, enter the root path where you installed your C/C++ compiler.
- 4 From the **HostTargetComm** list, choose **RS232**.
- 5 From the **RS232HostPort** list, choose either **COM1** or **COM2** for the connection on the host PC. Then, xPC Target automatically determines the COM port you use on the target computer.
- 6 When you finish changing the properties, click the **Update** button.

xPC Target updates the environment with the new properties.

You do not have to exit and restart MATLAB after making changes to the xPC Target environment, even if you change the communication between the host and target from RS232 to TCP/IP. However, you have to recreate the target boot disk, and rebuild the target application from the Simulink model.

For more information on the xPC Target Environment, see “Environment Reference” on page 6-1

Your next task is to create a target boot disk. See “Target Boot Disk” on page 2-23.

Network Communication

Before you can create and run a target application, you need to set up the connection between the host and target computers. You can use either serial or network communication.

This section includes the following topics:

- **“Hardware for Network Communication”**
- **“Ethernet Card for a PCI-Bus”**
- **“Ethernet Card for an ISA-Bus”**
- **“Environment Properties for Network Communication”**

For using serial communication, see “Serial Communication” on page 2-12.

Advantages of Network Communication - A host-to-target connection using network TCP/IP communication has advantages over serial RS232 communication:

- **Higher data throughput** - Network communication using Ethernet can transfer data up to 10 Mbit/s instead of the maximum data transfer rate of 115 kBaud with serial communication.
- **Longer distances between host and target computer** - By using repeaters and gateways you do not restrict the distance between your host and target computers to the length of a serial cable. Also, communication over the Internet is possible.

This manual does not include information for installing network cards or the TCP/IP protocol on your host computer. For correct installation and setup of your network cards and the TCP/IP protocol, contact your system administrator.

Hardware for Network Communication

You must install the following hardware before you install the xPC Target software and configure it for network communication:

- **Network adapter card** - When using xPC Target with TCP/IP, you must have a network adapter card correctly installed on both your host PC and your target PC. Then, you connect the host and target computers with a coaxial cable or unshielded twisted pair (UTP) cable to your local area network (LAN).

Also, you can directly connect your computers together. Use a cross-over UDP cable with RJ45 connectors, or add BNC T-connectors to your Ethernet boards, connect with a normal coaxial cable, and add BNC-terminators at the ends.

- **I/O boards** - If you use I/O boards on your target PC, you need to correctly install the boards.

Supported Ethernet cards - xPC Target includes device drivers for specific network cards. The MathWorks supplies a PCI-bus Ethernet card with the xPC Target software for you to use in your target PC. This card is NE2000 compatible.

Note The Ethernet card included with xPC Target supports a data transfer rate of 10 Mbit/s. We do not support a 100 Mbit/s Ethernet card.

The following are cases where you cannot use the Ethernet card we provide with xPC Target:

- You do not have an available PCI slot in your target PC
- You do not have a PCI-bus in your target PC
- You need to use an Ethernet card other than the card we provide with xPC Target

If one of the above cases applies, purchase one of the boards from the following list. We have tested these boards to be compatible with xPC Target.

Board Type	Board Number	xPC Target Driver
PCI	SMC EZ Card 10 SMC1208T (RJ45)	NE2000
	SMC EZ Card 10 SMC1208BT (RJ45, BNC)	NE2000
	SMC EZ Card 10 SMC1208BTA (RJ45, BNC, AUI)	NE2000
ISA	SMC EZ Card 10 SMC1660T (RJ45)	NE2000
	SMC EZ Card 10 SMC1660BT (RJ45, BNC)	NE2000
	SMC EZ Card 10 SMC1660BTA (RJ45, BNC, AUI)	NE2000
PC/104	Real Time Devices USA CM202 (RJ45, BNC, AUI)	NE2000
	WinSystems Inc. PCM-NE2000-16 (RJ45)	NE2000
	WinSystems Inc. PCM-NE2000-16-BNC (BNC)	NE2000
SBC	Versalogic VSBC-6	SMC91C9X

Ethernet Card for a PCI-Bus

If your target PC has a PCI-bus, we recommend that you use an Ethernet card for the PCI-bus. The PCI-bus has a faster data transfer rate and requires minimal effort to configure. Also, The MathWorks supplies one PCI-bus Ethernet card with the xPC Target software for your target PC.

To install the PCI-bus Ethernet card supplied with the xPC Target software, use the following procedure:

- 1 Turn off your target PC.
- 2 If the target PC already has an unsupported Ethernet card, remove the card.
- 3 Plug the Ethernet card from The MathWorks into a free PCI-bus slot.
- 4 Connect your target PC Ethernet card to your LAN using a coaxial cable or an unshielded twisted-pair cable.

Your next task is to set up the xPC Target environment for network communication. See “Environment Properties for Network Communication” on page 2-20.

Ethernet Card for an ISA-Bus

Your target PC might not have an available PCI-bus slot, or your target PC might not contain a PCI-bus (older motherboards, passive ISA-backplanes, or PC/104 computers). In these cases, you can use an Ethernet card for an ISA-bus.

If you are using an ISA-bus, you need to reserve, from the BIOS, an interrupt for this board.

The MathWorks does not provide an ISA-bus board. For a list of known compatible network adapter cards, see “Hardware for Network Communication” on page 2-16.

To install an ISA-bus Ethernet card, use the following procedure:

- 1 Turn off your target PC.
- 2 On your ISA-bus card, assign an IRQ and I/O-port base address by moving the jumpers or switches on the card. Write down these settings because you need to enter them in the xPC Target Setup window.

We recommend setting the IRQ line to 11 and the I/O-port base address around 0x300. If one of these hardware settings would lead to a conflict in your target PC, choose another IRQ or I/O-port base address.

Note If your ISA-bus card does not contain jumpers to set the IRQ line and the base address, use the utility on the installation disk supplied with your card to manually assign the IRQ line and base address. Do not configure the card as a PnP-ISA device.

- 3 If the target PC already has an unsupported Ethernet card, remove the card. Plug the compatible network card into a free ISA-bus slot.
- 4 Connect the target PC network card to your local area network (LAN) using a coaxial or an unshielded twisted-pair cable.

If you use an Ethernet card for an ISA-bus within a target PC with a PCI-bus, you must reserve the chosen IRQ line number for the Ethernet card in the PCI-BIOS. Refer to your BIOS setup documentation to setup the PCI-BIOS.

Your next task is to set up the xPC Target environment for network communication. See “Environment Properties for Network Communication” on page 2-20.

Environment Properties for Network Communication

The xPC Target environment is defined by a group of properties. These properties give xPC Target information about the software and hardware that it works with. You change some of these properties often while others you change only rarely.

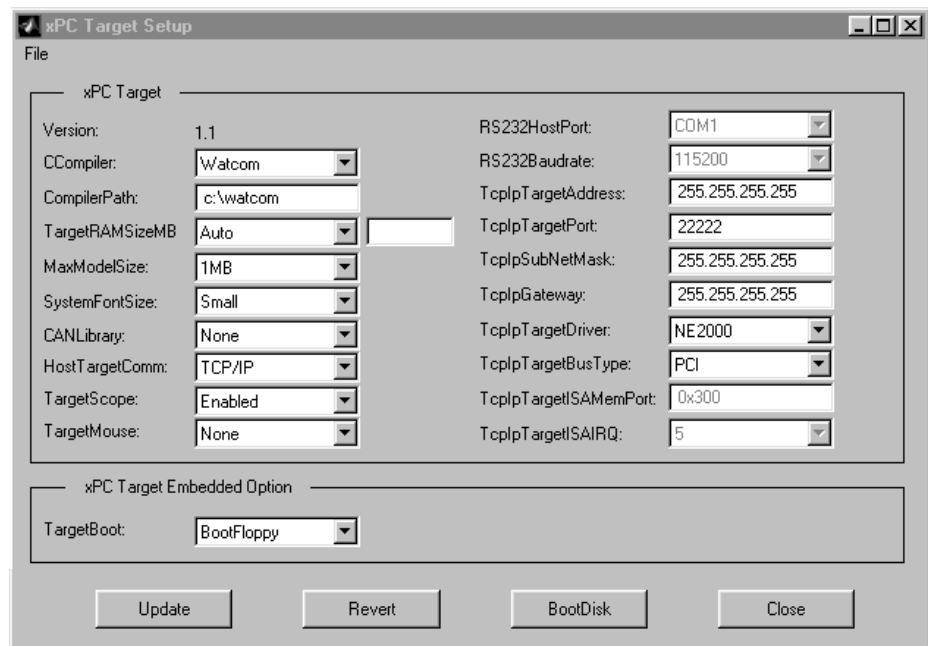
After you have installed xPC Target, you can set the specific environment properties for your host and target computers. You must change these environment properties before you can build and download a target application.

To change the environment properties using a graphical user interface, use the following procedure.

- 1 In the MATLAB window, type

`xpcsetup`

The xPC Target Setup window opens.



The xPC Target Setup window has two sections:

- xPC Target
- xPC Target Embedded Option

If your license does not include the embedded option, the **TargetBoot** list is disabled (grayed-out) with **Boot Floppy** as your only choice. With the xPC Target Embedded Option installed, you have the additional choices of **DOSLoader** and **StandAlone**.

- 2 From the **CCompiler** list, choose either **VisualC** or **Watcom**.
- 3 In the **CompilerPath** box, enter the root path where you installed the your C/C++ compiler.
- 4 From the **HostTargetComm** list, choose **TCP/IP**.

The TCP/IP text boxes become active.

You must enter the following properties with the correct values according to your LAN environment. Ask your system administrator for values to the following settings:

- **TcpIpTargetAddress** - This is the IP address for your target PC. An example of an IP address is 192. 168. 0. 1.
- **TcpIpSubNetMask** - This is the Subnet Mask address of your LAN. An example of a Subnet Mask address is 255. 255. 0. 0.

You enter the following properties depending on your specific circumstances:

- **TcpIpTargetPort** - This property is set by default to 22222. This value should not cause any problems, because this number is higher than the reserved area (telnet, ftp, ...) and it is only of relevance on the target PC. If necessary this property value can be changed to any value higher than 20000.
- **TcpIpGateway** - This property is set by default to 255. 255. 255. 255. This means that your do not use a gateway to connect to your target PC. If you communicate with the target PC from within your LAN, you may not need to define a gateway and change this setting.

If you communicate from a host PC located in a LAN different from your target PC you need to define a gateway and enter its IP address. This is especially true if you want to work over the Internet. Ask your system administrator for the IP address of the appropriate gateway.

The following properties are specific for the Ethernet card on your target PC:

- **TcpIpTargetDriver** - This property is set by default to NE2000. xPC Target also supports the SMC91C9X drivers.
- **TcpIpTargetBusType** - This property is set by default to PCI. If TcpIpTargetBusType is set to PCI, then the properties TcpIpISAMemPort and TcpIpISAIRQ have no effect on TCP/IP communication and are disabled (grayed out). If you are using an ISA-bus Ethernet card, set TcpIpTargetBusType to **ISA** and enter values for TcpIpISAMemPort and TcpIpISAIRQ.
- **TcpIpISAMemPort and TcpIpISAIRQ** - If you are using an ISA-bus Ethernet card, you must enter values for the properties TcpIpISAMemPort and TcpIpISAIRQ. The values of these properties must correspond to the jumper settings or ROM settings on your ISA-bus Ethernet card.

5 When you finish changing the properties, click the **Update** button.

xPC Target updates the environment with the new properties.

You do not have to exit and restart MATLAB after making changes to the xPC environment even, if you changed the communication between the host and target from RS232 to TCP/IP. Only the SIMULINK model has to be rebuilt.

For more information on the xPC Target Environment, see “Environment Reference” on page 6-1.

Your next task is to create a target boot disk. See “Target Boot Disk” on page 2-23.

Target Boot Disk

You use the target boot disk to load and run the xPC Target kernel

After you make changes to the xPC Target environment properties, you need to create or update a target boot disk.

To create a target boot disk for the current xPC Target environment, use the procedure below:

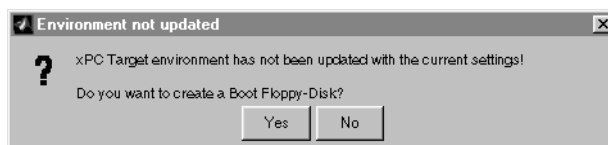
- 1 In the MATLAB window, type

```
xpcsetup
```

The xPC Target Setup window opens.

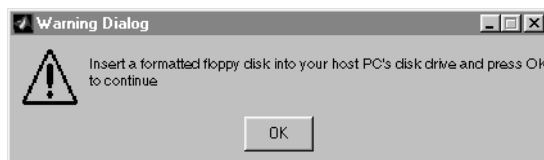
- 2 Click the **BootDisk** button.

If you didn't update the current settings, the following message box opens.



Click **No**. Click the **Update** button, and then click the **BootDisk** button again.

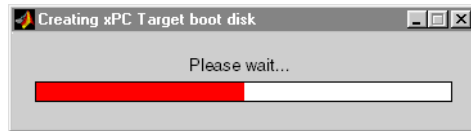
After you update the current properties, and click the **BootDisk** button, the following message box opens.



- 3 Insert a formatted 3 1/2 inch floppy disk into the host PC disk drive, and then click **OK**.

Note All data on the disk will be erased.

xPC Target displays the following dialog box while creating the boot disk. The process takes about 1 to 2 minutes.



- 4 Close the xPC Target Setup window.
- 5 Remove the target boot disk from the host PC disk drive and insert it into your target PC disk drive.

Your next task is to test your installation. See “Testing and Troubleshooting the Installation” on page 2-26.

Current Properties on the Target Boot Disk

To check if a target boot disk corresponds to the current xPC Target environment, use the following procedure.

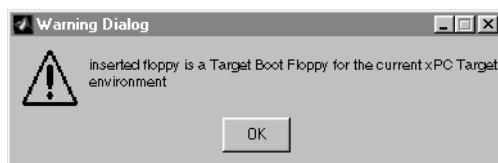
To check the target boot disk with an xPC Target GUI

- 1 Insert the target boot disk into your host PC drive, and type
`xpcsetup`

The xPC Target Setup window opens

- 2 In the Setup window, click the **BootDisk** button.

If the boot disk properties correspond to the current properties, the following message opens and no data is written to the disk.



To check the target boot disk with MATLAB commands

- 1 Insert the target boot disk into your host PC drive, and type
`xpcbootdisk`

MATLAB displays the message

Insert a formatted floppy disk into your host PC's disk drive and press a key to continue

- 2 Press any key.

If the boot disk properties correspond to the current properties, MATLAB displays the following message and not data is written to the disk.

Inserted floppy is a Target Boot Floppy for the current xPC Target environment

Testing and Troubleshooting the Installation

Use this section to troubleshoot connection and communication problems between your host and target computers.

This section includes the following topics:

- “Testing the Installation”
- “Test 1, Ping Target System Standard Ping”
- “Test 2, Ping Target System xPC Target Ping”
- “Test 3, Reboot Target Using Direct Call”
- “Test 4, Build and Download Application”

Testing the Installation

xPc Target uses a function to test the entire installation. After you install the xPC Target software, set the environment settings, and create a target boot disk, you can test your installation.

- 1 Insert your target boot disk into your target PC disk drive.
- 2 Press the reset button on your target PC.

After loading the BIOS, xPC Target boots the kernel, and displays the following screen on the target PC.

Loaded App: 1MB free Memory: 61MB Mode: loader Logging: - StopTime: - SampleTime: - AverageTET: - Execution: -	<pre> ----- * xPC Target 1.0, (c) 1996-99 The MathWorks Inc. * ----- System: Host-Target Interface is RS232 (COM1/2) System: COM1 detected </pre>
---	---

- 3 In the MATLAB window, type

xpctest

MATLAB runs the test script and displays messages indicating the success or failure of a test. If you use RS232 communication, the first test is skipped.

```
### xPC Target Test Suite 1.1
### Host-Target interface is: TCP/IP (Ethernet)
### Test 1, Ping target system using standard ping: ... OK
### Test 2, Ping target system using xpctargetping: ... OK
### Test 3, Reboot target using direct call: ..... OK
### Test 4, Build and download xPC Target application: ... OK
### Test 5, Check host-target communication for commands: .. OK
### Test 6, Download xPC Target application using OOP: ... OK
### Test 7, Execute xPC Target application for 0.2s: ... OK
### Test 8, Upload logged data and compare with simulation: . OK
### Test Suite successfully finished
```

If any of the tests fail, see the appropriate test section:

- “Test 1, Ping Target System Standard Ping”
- “**Test 2, Ping Target System xPC Target Ping**”
- “**Test 3, Reboot Target Using Direct Call**”
- “**Test 4, Build and Download Application**”

Test 1, Ping Target System Standard Ping

If you are using a network connection, this is a standard system ping to your target computer. If this test fails, try troubleshooting with the following procedure:

- 1 Open a DOS shell, and type the IP address of the target computer

```
ping xxx. xxx. xxx. xxx
```

DOS should display a message similar to the following.

```
Pinging xxx. xxx. xxx. xxx with 32 bytes of data:
Reply from xxx. xxx. xxx. xxx: bytes= 32 time<10nx TTL=59
```

2 Check the messages on your screen.

Ping command fails - If the DOS shell displays the following message.

```
Pinging xxx.xxx.xxx.xxx with 32 byte of data:  
Request timed out.
```

The ping command failed, and the problem may be with your network cables.

To solve this problem, check your network cables. You may have a faulty network cable, or if you are using a coaxial cable, the terminators may be missing.

Ping command fails, but cables are okay - If the cables are okay, the problem may be that you entered an incorrect property in the Setup window.

To solve this problem, in the MATLAB command window, type

```
xpcsetup
```

Check that **TcpIpTargetAddress**, **TcpIpSubNetMask**, and **TcpIpGateway** have the correct values.

For a PCI-bus:

- Check that **TcpIpTargetBusType** is set to PCI instead of ISA.

For an ISA-bus:

- Check that **TcpIpTargetBusType** is set to ISA instead of PCI.
- Check that **TcpIpTargetISAMemPort** is set to the correct I/O-port base address, and check that the address does not lead to a conflict with another hardware resource.
- Check that **TcpIpTarget IRQ** is set to the correct IRQ line, and check that the line number does not lead to a conflict with another hardware resource.
- If the target PC motherboard contains a PCI chipset, check if the IRQ line used by the ISA-bus Ethernet card is reserved within the BIOS setup.

Ping succeeds, but test 1 with the command `xpc test` fails - The problem may be that you have incorrect IP and gateway addresses entered in the Setup window.

To solve this problem, in the MATLAB command window, type

```
xpcsetup
```

Enter the correct addresses. Click the **Update** button. Recreate the target boot disk by inserting a floppy disk into the host disk drive, and then clicking the **BootDisk** button.

If you still cannot solve your problem, see “If You Still Need More Help” on page 2-31.

Test 2, Ping Target System xPC Target Ping

This test is an xPC Target ping to your target computer. If this test fails, try troubleshooting with the following procedure.

- 1 In the MATLAB window, type

```
tg=xpc
```

- 2 Check the messages in the MATLAB window.

MATLAB should respond with the following messages.

```
xPC Object
Connected          = Yes
Application        = loader
```

Target object does not connect - If you do not get the above messages, the problem may be that you have a bad target boot disk.

To solve this problem, create another target boot disk with a new floppy disk. See “Target Boot Disk” on page 2-23.

If you still cannot solve your problem, see “If You Still Need More Help” on page 2-31.

Test 3, Reboot Target Using Direct Call

This test tries to boot your target computer using an xPC Target command. If this test fails, try troubleshooting with the following procedure.

- 1 In the MATLAB window, type

```
xpctest noreboot
```

This command reruns the test without using the reboot command and displays the message

```
### Test 3, Reboot target using direct call: ... SKIPPED
```

- 2 Observe the messages in the MATLAB command window during the build process.

Reboot fails, but build okay when reboot skipped - If the command `xpctest` skips the reboot command, and successfully builds and loads the target application, the problem is some target hardware does not support the xPC Target reboot command. In this case, you cannot use this command to reboot your target computer. You need to reboot using a hardware reset button.

If you still cannot solve your problem, see “If You Still Need More Help”.

Test 4, Build and Download Application

This test tries to build and download the model `xpcosc.mdl`. If this test fails, try troubleshooting with the following procedure:

- 1 In the MATLAB command window, check the error messages.

These messages help you locate where there is a problem.

- 2 If you get the error message

```
xPC Target loader not ready
```

Reboot your target computer. This error message is sometimes displayed even if the target screen shows the loader is ready.

If you still cannot solve your problem, see “If You Still Need More Help”.

If You Still Need More Help

If you cannot solve your problem, contact The MathWorks directly for help.

Internet <http://www.mathworks.com/support>

E-mail <mailto:support@mathworks.com>

Telephone 508-647-7000

Ask for Technical Support

Basic Procedures

Simulating the Model	3-3
Loading a Simulink Model	3-3
Running a Simulation Using the Simulink Graphical Interface	3-4
Running a Simulation Using the MATLAB Command Line Interface	3-5
Creating the Target Application	3-7
Booting the Target PC	3-7
Troubleshooting the Boot Process	3-8
Entering the Simulation Parameters	3-8
Building and Downloading the Target Application	3-13
Troubleshooting the Build Process	3-15
Controlling the Target Application	3-16
Control with MATLAB Commands	3-17
Signal Monitoring	3-19
Signal Monitoring with MATLAB Commands	3-19
Signal Logging	3-20
Signal Logging with xPC Target Graphical Interface	3-20
Signal Logging with MATLAB Commands	3-22
Signal Tracing	3-26
Signal Tracing with xPC Target GUI	3-26
Signal Tracing with xPC Target GUI (Target Manager)	3-31
Signal Tracing with MATLAB Commands	3-35
Parameter Tuning	3-38
Parameter Tuning with MATLAB Commands	3-38
Parameter Tuning with Simulink External Mode	3-40

The procedures in this chapter explain the basic functions of xPC Target by using a simple Simulink model. The model is an oscillator with a square wave input. Since this model does not have I/O blocks, you can try these procedures regardless of whether you have I/O hardware on the target PC.

This chapter includes the following sections:

Simulink Model

- **“Simulating the Model”** - Run a simulation of your Simulink/Stateflow model on the host PC.

xPC Target Application

- **“Creating the Target Application”** - Use Real-Time Workshop, Stateflow Coder, and a C compiler to create a real-time application.
- **“Controlling the Target Application”** - Run the target application in real time on the target PC.

Signal Acquisition and Analysis

- **“Signal Monitoring”** - Get signal data without time information while running your real-time application.
- **“Signal Logging”** - Save signal data for analysis after completing a real-time run.
- **“Signal Tracing”** - Visualize signals while running your real-time application.

Parameter Tuning

- **“Parameter Tuning”** - Change block parameters while running your real-time application.

Simulating the Model

You use Simulink in normal mode to observe the behavior of your model in nonreal-time. This section includes the following topics:

- **“Loading a Simulink Model”**
- **“Running a Simulation Using the Simulink Graphical Interface”**
- **“Running a Simulation Using the MATLAB Command Line Interface”**

For procedures to run your target application in real-time, see “Creating the Target Application” on page 3-7.

Loading a Simulink Model

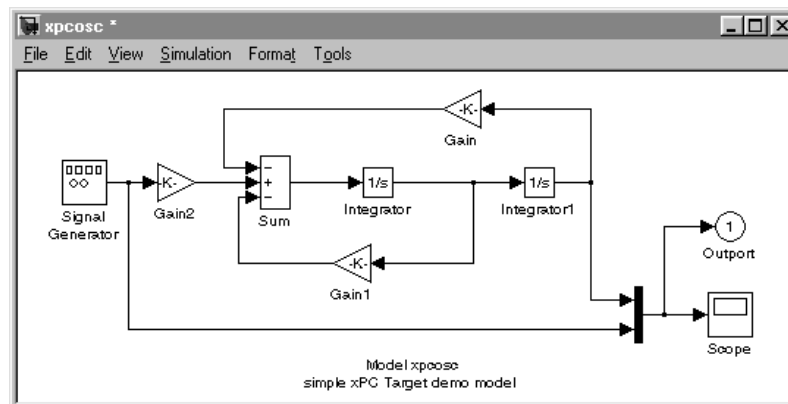
Loading a Simulink model moves information about your model, including the block parameters, into the MATLAB workspace.

After you create and save a Simulink model, you can load it back into the MATLAB workspace. This procedure uses the Simulink model `xpcosc.mdl` as an example.

- 1 In the MATLAB window, type

```
xpcosc
```

MATLAB loads the oscillator model and displays the Simulink block diagram, as shown below.



Your next task is to run a simulation of your model in nonreal-time. See “Running a Simulation Using the Simulink Graphical Interface” or “**Running a Simulation Using the MATLAB Command Line Interface**”.

Running a Simulation Using the Simulink Graphical Interface

You run a simulation of your model to observe the behavior of the 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 `xpcosc.mdl` as an example and assumes you have already loaded that model.

- 1 From the **Simulation** menu, click **Normal**, and then click **Start**.
- 2 In the Simulink window, double-click the output scope block.

Simulink opens a scope window showing the output of the model.



- 3 You can either let the simulation run to its stop time, or stop the simulation manually. To stop the simulation manually, from the **Simulation** menu, click **Stop**.

Running a Simulation Using the MATLAB Command Line Interface

You run a simulation of your model to observe the behavior of the 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 `xpcosc.mdl` as an example and assumes you have already loaded that model.

- 1 In the MATLAB window, type

```
sim('xpcosc')
```

Simulink runs a simulation in normal mode.

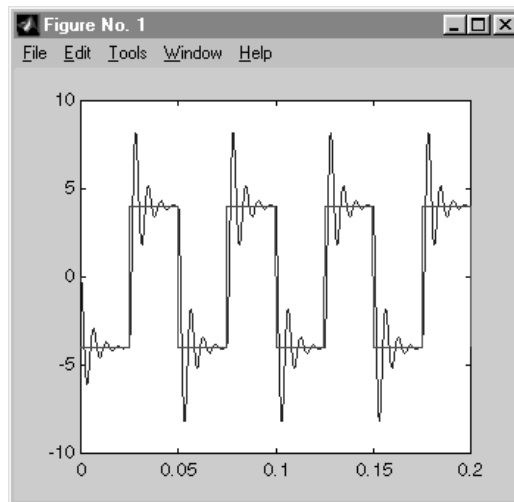
- 2 You can either let the simulation run to its stop time, or stop the simulation manually. To stop the simulation manually, press Ctrl-T.

- 3 After Simulink finishes the simulation, type

```
plot(tout, yout)
```

You enter the MATLAB variables `tout` and `yout` in the Simulation Parameters dialog box. The signals are logged into memory through Outport blocks. See “Entering the Simulation Parameters” on page 3-8.

MATLAB opens a plot window and displays the output response. The signal from the signal generator is added to the outport block and shown in the figure below.



Note When running your target application in real-time, data is not saved to the variable `tout` and `yout`. Instead, data is saved to the target object properties `TimeLog`, `StateLog`, and `OutLog`. However, you must still select the **Time**, **States**, and **Output** check boxes for data to be logged into the target object properties.

Your next task is to create a target application. See “Creating the Target Application” on page 3-7.

Creating the Target Application

You run the target application to observe the behavior of your model in real-time.

This section includes the following topics:

- **“Booting the Target PC”**
- **“Troubleshooting the Boot Process”**
- **“Entering the Simulation Parameters”**
- **“Building and Downloading the Target Application”**
- **“Troubleshooting the Build Process”**

For procedures to simulate your model in nonreal-time, see “Simulating the Model” on page 3-3.

Booting the Target PC

Booting the target computer loads and starts the xPC Target kernel on the target PC. The loader then waits for xPC Target to download your target application from the host PC.

After you have configured xPC Target using the Setup window, and created a target boot disk for that setup, you can boot the target PC. You need to boot the target computer before building your target application because the build process automatically downloads your target application to the target PC.

- 1 Insert the target boot disk into the target PC disk drive.
- 2 Turn on the target PC or press the reset button.

The target PC displays the following screen.

<pre> Loaded App: 1MB free Memory: 61MB Mode: loader Logging: - StopTime: - SampleTime: - AverageTET: - Execution: - </pre>	<pre> ----- * xPC Target 1.0, (c) 1996-99 The MathWorks Inc. * ----- System: Host-Target Interface is RS232 (COM1/2) System: COM1 detected </pre>
--	---

In the example above, the status window shows the kernel is in the loader mode and waiting to load a target application. 1MB of memory is reserved for the

application, 3 MB is used by the kernel, and 61 MB is available from a total of 64 MB. The xPC Target kernel uses the 61 MB for the heap, running scopes, and acquiring data.

Your next task is to enter the simulation and real-time run parameters for Real-Time Workshop. See “Entering the Simulation Parameters” on page 3-8

Troubleshooting the Boot Process

Possible Problem. When booting the target PC, it might display the message:

```
xPC Target 1.1 loading kernel . . @@@@
```

The target PC displays this message when it cannot read and load the kernel from the target boot disk. The probable cause is a bad disk.

Solution. The solution is to reformat the disk or use a new preformatted floppy disk and create a new target boot disk.

Entering the Simulation Parameters

The simulation and real-time run parameters are entered in the Simulink Parameters dialog box. They give information to Real-Time Workshop for how to build the target application from your Simulink model.

After you load a Simulink model and boot the target PC, you can enter the simulation parameters. This procedure uses the Simulink model `xpcosc.mdl` as an example and assumes you have already loaded that model.

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

Simulink displays the Solver page. This page defines the initial stop and sample time for your target application.

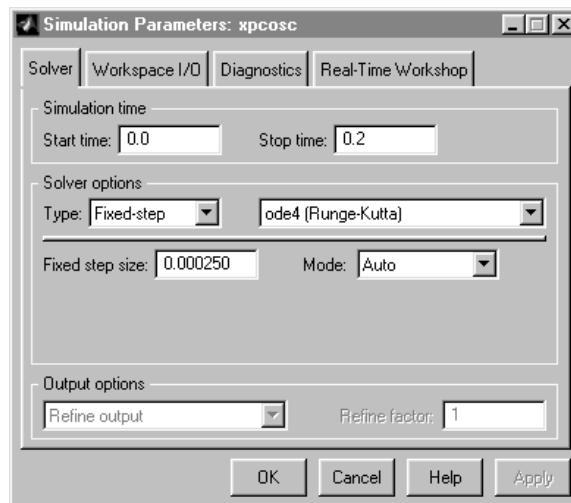
- 2 In the **Start time** box, enter **0** seconds. In the **Stop time** box, enter and initial stop time. For example, enter **0.2** seconds. You can change this time after creating your target application by changing the target object property `tg.Stoptime`.
- 3 From the **Type** list, choose **Fixed-step**. Real-Time Workshop does not support variable step solvers. From the integration algorithm list choose a solver. For example, choose the general purpose solver **ode4 (Runge-Kutta)**.

In the **Fixed step size** box, enter the sample time for the target application. For example, enter 0.00025 seconds (250 microseconds). You can change this value after creating the target application.

If you find that 0.000250 seconds results in overloading the CPU on the target PC, try a larger **Fixed step size** such as 0.002 seconds.

If your model contains discrete states, which would lead to a hybrid model, the sample times of the discrete states can only be multiples of the **Fixed step size**. If your model does not contain any continuous states, enter 'auto', and the sample time is taken from the model.

The Solver page should look similar to the figure shown below.



- 4 In the **Simulation Parameters** dialog box, click the **Workspace I/O** tab.

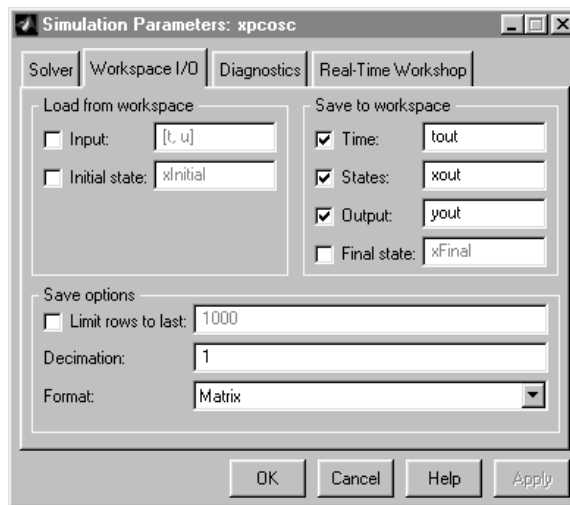
The Workspace I/O page opens. This page defines which model signals are logged during the simulation of your model or the real-time run of your target application.

- 5 In the **Save to workspace** section, choose the **Time**, **State**, and **Output** check boxes.

To save (log) data from signals other than the state values, you need to add outputport blocks to your Simulink model and connect the signals to the outputport blocks.

Note When running your target application in real-time, data is not saved to the variables `tout` and `yout`. Instead, data is saved to the target object properties `TimeLog`, `StateLog`, and `OutLog`. However, you must still select the **Time**, **States**, and **Output** check boxes for data to be logged into the target object properties.

The Workspace I/O page should look similar to the figure shown below.



Normally all the check boxes are activated, except maybe in the following cases:

- **Many states** - If your model contains many states (for example, greater than 20 states) the storage of the state vector requires a lot of target memory. By turning off logging of the state signals, more memory is available for the target application. An alternative to logging all of the state signals is for you to select individual states of interest by adding outputport blocks to your model.

- **Small sample time** - You may choose a very short sample time that overloads the CPU. By turning off certain logging options more computing time is available for calculating the model.
- 6 In **Simulation Parameters** dialog box, click the **Real-Time Workshop** tab.

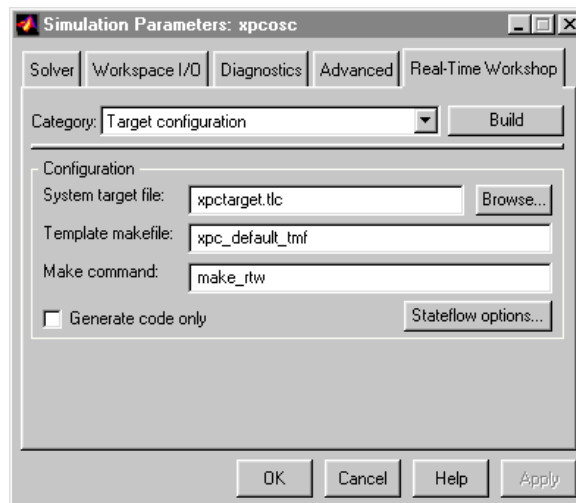
The Real-Time Workshop page opens.

- 7 Click the **Browse** button.

The System Target File Browser opens.

- 8 From the list, choose **xpctarget.tlc xPC Target**. Click **Ok**.

The System target file `xpctarget.tlc`, the Template makefile `xpc_default_tmf`, and the Make command `make_rtw` are automatically entered into the page. The Real-Time Workshop page should now look like the figure shown below.



- 9 From the Category list, choose **xPC Target code generation options**.

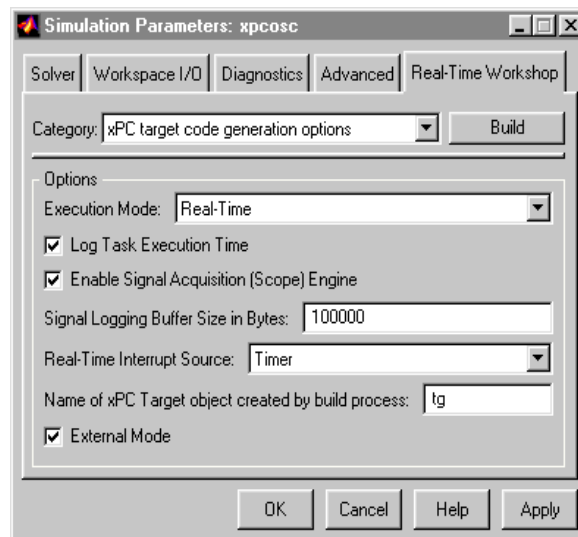
The Real-Time Workshop page opens to the options page.

- 10 From the Mode list, choose either **Real-Time** or **Freerun**. **Freerun** is similar to a simulation, but with the generated code.
- 11 Select the **Log Task Execution Time** check box to log the task execution time to the target object property t.g. TETlog. Selecting the Enable Signal Acquisition (Scope) Engine check box allows you to add scopes to the target PC.
- 12 In the **Signal Logging Buffer Size in Bytes** box, enter the maximum number of sample points to save before wrapping. This buffer includes the time, states, outputs, and task execution time logs.

For example, the model xpcosc.mdl has 6 signals (1 time, 2 states, 2 outputs, and 1 TET). If you enter a buffer size of 100000, then the target object property t.g. MaxLogSample is calculated as $100000 / 6 = 1666$. After saving 1666 sample points the buffer wraps to collect the next 1666 samples.

If you select a logging buffer size larger than the RAM on the target PC, the target PC displays a message, ERROR: allocation of logging memory failed, after downloading and initializing the target application. In this case you need to install more RAM or reduce the buffer size for logging. In any case the target PC has to be rebotted.
- 13 From the **Interrupt Source** list, select a source. The default value is set to **Timer**.
- 14 In the **Name of xPC Target object** box, enter the a target object name. The default target object name is t.g.

The options page should now look similar to the figure shown below.



15 Click **OK**.

Your next task is to create (build) the target application. See “Building and Downloading the Target Application” on page 3-13.

Building and Downloading the Target Application

You use the xPC Target build process to generate C code, compile, link, and download your target application to the target PC.

After you enter your changes in the **Simulation Parameters** dialog box, you can build your target application. This procedure uses the Simulink model `xpcosc.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**. From the **Real-Time Workshop** submenu, click **Build Model**.

After the compiling, linking, and downloading process, a target object is created with properties and associated methods. The default name of the

target object is `tg`. For more information about the target object, see “Target Object Reference” on page 7-1.

On the host computer, MATLAB displays the following lines after a successful build process.

```
### Starting Real-Time Workshop build procedure for model:
xpcosc
. . .
### Successful completion of xPC Target build procedure for
model: xpcosc
```

The target PC displays the following information.

Loaded App: xpcosc	* xPC Target 1.0, (c) 1996-99 The MathWorks Inc. *
Memory: 61MB	-----
Mode: RT, single	System: Host-Target Interface is RS232 (COM1/2)
Logging: t x y tet	System: COM1 detected
StopTime: 0.2 d	System: download started...
SampleTime: 0.00025	System: download finished
AverageTET: -	System: initializing application...
Execution: stopped	System: initializing application finished

2 In the MATLAB window, type

```
tg
```

MATLAB displays a list of properties for the target object `tg`.

If you do not have a successful build, see “Troubleshooting the Build Process” on page 3-15.

Your next task is to run the target application in real-time on the target PC. See “Controlling the Target Application” on page 3-16.

Troubleshooting the Build Process

If the host PC and target PC are not properly connected or you have not correctly entered the environment properties, the download process is terminated after about 5 seconds with a time-out error.

To correct the problem use the following procedure.

- 1 In the MATLAB window, type

```
xpcsetup
```

The xPC Target Setup window opens.

- 2 Check and, if necessary, make changes to the communication properties, update the properties, and recreate the target boot disk.

For information on the procedures, see either “Environment Properties for Serial Communication” on page 2-13 or “Environment Properties for Network Communication” on page 2-20, and then see “Target Boot Disk” on page 2-23.

Controlling the Target Application

During the build process, a target object was created that represents the target application and the target computer. The target object is defined by a set of properties and associated methods. You control the target application and computer by changing the target object properties with the target object methods.

This section includes the following topic:

- **“Control with MATLAB Commands”**

For controlling the target application from:

- The target PC, see “Target PC Command Line Interface” on page 4-19.
- A Web browser, see “Web Interface” on page 4-25.

For more information about the target object properties and methods, see “Target Object Reference” on page 7-1.

Control with MATLAB Commands

You run your target application in real time to observe the behavior of your model with generated code.

After xPC Target downloads your target application to the target PC, you can run the target application. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model. The default name of the target object is `tg`.

1 In the MATLAB window, type

```
+tg or tg.start or start(tg)
```

The target application starts running on the target PC. In the MATLAB window, the status of the target object changes from stopped to running.

```
xPC Object
Connected          = Yes
Application         = xpcosc
Application         = xpcosc
Mode               = Real - Time Single-Tasking
Status             = running
```

On the target PC screen, the Simulation line changes from stopped to running and the AverageTET line periodically updates with a new value.

Loaded App: xpcosc	System: COM1 detected
Memory: 61MB	System: download started...
Mode: RT, single	System: download finished
Logging: t x y tet	System: initializing application...
StopTime: 0.2 d	System: initializing application finished
SampleTime: 0.00025	System: execution started (sample time: 0.000250)
AverageTET: 8.608e-006	System: execution stopped at 0.200000
Execution: stopped	minimal TET: 0.000008 at time 0.003500
	maximal TET: 0.000011 at time 0.000500

2 In the MATLAB window, type

```
-tg or tg.stop or stop(tg)
```

The target application stops running.

xPC Target allows you to change many properties and parameters without rebuilding your target application. Two of these properties are stop time and sample time.

After you build a target application, but before you start running the application you can change the sample time. You can change the stop time before you start the target application or while it is running.

- 1 Change the stop time. For example, to change the stop time to 1000 seconds, type either

```
tg.StopTime = 1000 or set(tg, 'StopTime', 1000)
```

- 2 Change the sample time. For example, to change the sample time to 0.01 seconds, type either

```
tg.SampleTime = 0.01 or set(tg, 'SampleTime', 0.01)
```

Although you can change the sample time in between different runs, you can only change the sample time without rebuilding the target application under certain circumstances.

If you choose a sample time that is too small, a CPU overload can occur. If a CPU overload occurs, the target object property `CPU Overload` changes to detected. In that case, change the **Fixed step size** in the Solver property sheet to a larger value.

Signal Monitoring

Signal monitoring is the process for acquiring signal data during a real-time run without time information. The advantage to signal monitoring is that there is no additional load to the real time tasks.

This section includes the following topic:

“Signal Monitoring with MATLAB Commands”

Signal Monitoring with MATLAB Commands

Use signal monitoring to acquire signal data without creating scopes that run on the target PC.

After you start running a target application, you can use signal monitoring to get signal data:

- 1 In the MATLAB window, type

```
tg.S1
```

MATLAB displays the value of the signal S1.

```
ans=  
3.731
```

Although, this procedure shows how to get signal data interactively, you would normally use signal monitoring in a MATLAB M-file script.

Signal Logging

Signal logging is the process for acquiring signal data during a real-time run, and after the run reaches its final time or after you manually stop the run, transferring the data to the host PC for analysis. You can plot the data, and later save it to a disk.

This section includes the following topics:

- “Signal Logging with xPC Target Graphical Interface”
- “Signal Logging with MATLAB Commands”

Signal Logging with xPC Target Graphical Interface

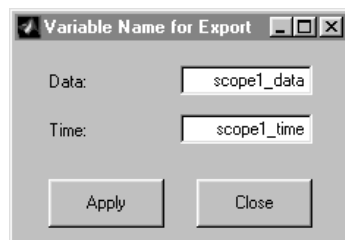
Exporting data with the xPC Target scope window does not require you to add outport blocks to your Simulink model and activate the logging of signals. You can select which signals to collect, and you can capture unexpected outputs during a run.

This procedure uses a scope window as a graphical interface to move data from the last data package collected to the MATLAB workspace. For information on exporting data using outport blocks, a scope object and the properties `Data` and `Time`, see Chapter 8, “Scope Object Reference.”.

After you complete a run with a scope, you can move data from the last data package collected to the MATLAB workspace. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have completed a run with the target application.

- 1 In the xPC Target Scope window, and from the **Plot** menu, click **Variable Name for Export**.

The **Variable Name for Export** dialog box opens.



- 2 In the **Data** and **Time** text boxes, enter the name of the MATLAB variables to contain the data from the scope data package. Click the **Apply** button, and then click the **Close** button.

The default name for the time vector is `scopen_time`, and the default name for the signal vector is `scopen_data` where `n` is the scope number.

- 3 In the Scope window, click the **Export** button. You can export data regardless of whether a scope is started or stopped.
- 4 In the MATLAB window, type

```
whos
```

MATLAB displays a list of variables and their description. For example

Name	Size	Bytes	Class
ans	1x1	13958	xpc object
scope1_data	250x2	4000	double array
scope1_time	250x1	2000	double array
t	801x1	6408	double array
tg	1x1	14002	xpc object
x	801x2	12816	double array
y	801x1	6408	double array

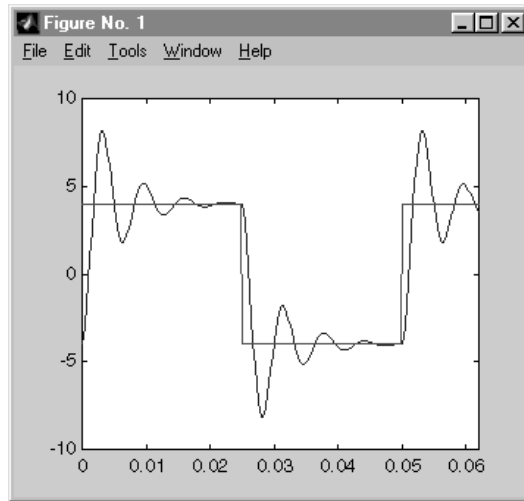
Grand total is 5828 elements using 59592 bytes

You can now save or further analyze the data using the MATLAB variables.

- 5 Type

```
plot(scope1_time, scope1_data)
```

MATLAB plots the variables `scope1_time` and `scope1_data` in a new window



Signal Logging with MATLAB Commands

You analyze and plot the outputs and states of your target application to observe the behavior of your model, or to determine the behavior when you vary the input signals.

Time, states and outputs. Logging the time, state, and output signals is possible only if you add, before the build process, outport blocks to your Simulink model, and in the **I/O-Workspace** page select the Save to workspace check boxes. See “Entering the Simulation Parameters” on page 3-8

Task execution time. Plotting the task execution time is possible only if you select the **Log Task Execution Time** check box in the **xPC Target code generation option page**. See “Entering the Simulation Parameters” on page 3-8.

After you run a target application, you can plot the state and output signals. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model.

- 1 In the MATLAB window, type either

```
+tg or tg.start or start(tg)
```

The target application starts and runs until it reaches the final time set in the target object property `tg.StopTime`.

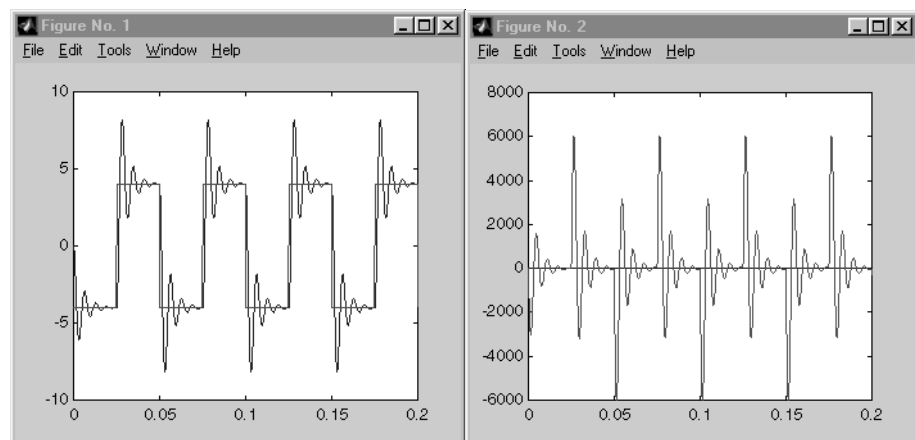
The outputs are the signals connected to Simulink output blocks. The model `xpcosc.mdl` has just one output block labeled **1** and there are two states. This output block shows the signals leaving the blocks labeled **Integrator1** and **Signal Generator**.

- 2 Plot the signals from the output block and the states. In the MATLAB window, type

```
plot(tg.TimeLog, tg.OutputLog)
figure
plot(tg.TimeLog, tg.StateLog)
```

Values for the logs are uploaded to the host PC from the target application on the target PC. If you want to upload part of the logs, see the target object method “`getlog`” on page 7-23.

The plots shown below are the result of a real-time execution. To compare this plot with a plot for a nonreal-time simulation, see “Running a Simulation Using the Simulink Graphical Interface” on page 3-4.



The task execution time (TET) is the time to calculate the signal values for the model during each sample interval. If you have subsystems that run only under certain circumstances, plotting the TET would show when subsystems were executed and the additional CPU time required for those executions.

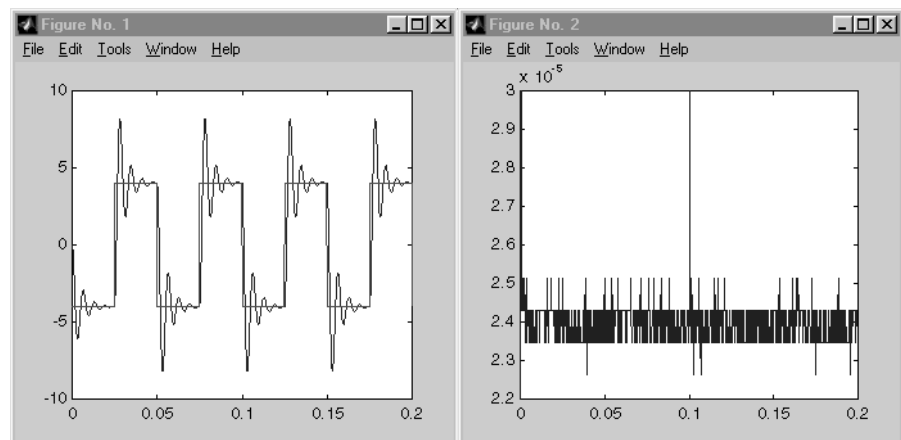
After you run a target application, you can plot the last execution time. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model.

- 1 Plot the signals from the output block and the task execution time (TET).
In the MATLAB window, type

```
plot(tg.TimeLog, tg.OutputLog)
figure
plot(tg.TimeLog, tg.TETLog)
```

Values for the logs are uploaded to the host PC from the target application on the target PC. If you want to upload part of the logs, see the target object method “getlog” on page 7-23.

The plots shown below are the result of a real-time run. The output is shown on the left and the task execution time is shown on the right.



- 2 Get information about the average task execution time. In the MATLAB window, type either

```
tg.AvgTET or get(tg, 'AvgTET')
```

MATLAB displays the following information.

```
ans =  
0.000009
```

In the example above, the minimum TET was 8 μ s, the maximum TET 11 μ s, and the average TET 9 μ s. This means that the real-time task has taken about 3 % of the CPU performance (Average TET of 9 μ s / Sample time of 250 μ s).

Signal Tracing

Signal tracing is the process of acquiring and visualizing signals during a real-time run. It allows you to acquire signal data and visualize it on your target PC or upload the signal data and visualize it on your host PC while the target application is running.

This section includes the following topics:

- “**Signal Tracing with xPC Target GUI**”
- “**Signal Tracing with xPC Target GUI (Target Manager)**”
- “**Signal Tracing with MATLAB Commands**”

Signal logging differs from signal tracing. With signal logging you can only look at a signal after a run is finished, and the entire log of the run is available. For information on signal logging, see “Signal Logging” on page 3-20.

Signal Tracing with xPC Target GUI

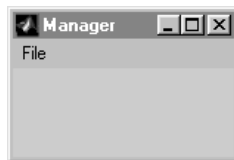
Opening an Scope window on the host PC allows you to view signals with a graphical user interface (GUI).

After you create, download, and start running a target application, you can view signals. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you are running the target application for that model.

- 1 In the MATLAB window, type

```
xpcscope
```

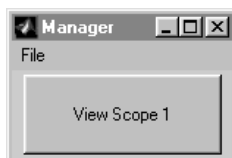
The Manager window opens. This window is the root-window of the xPC Target Scope graphical interface.



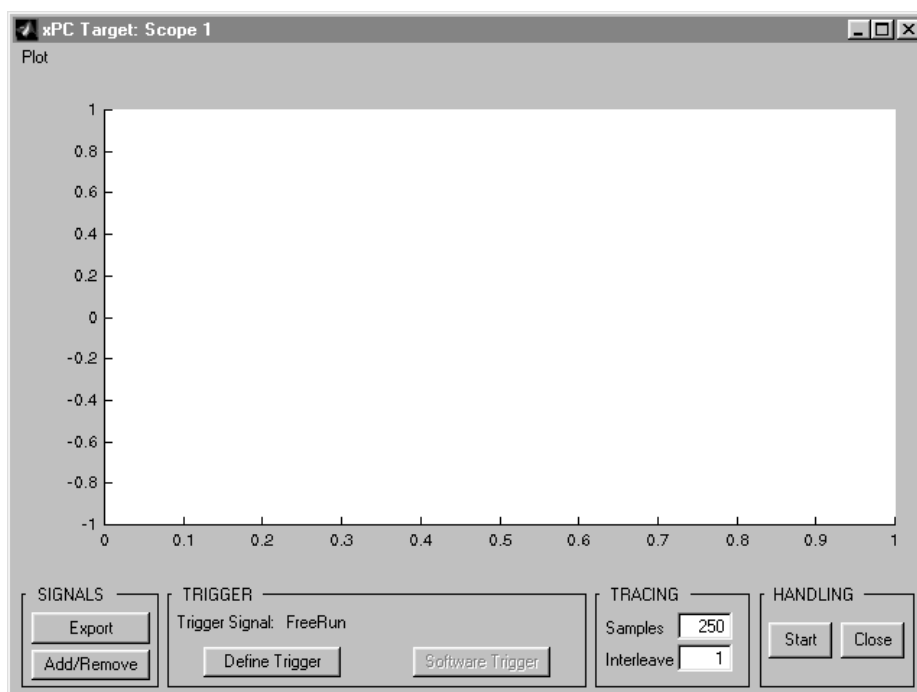
At this point, the window is empty because you need to define a specific scope.

- 2 From the **File** menu, click **New Scope**.

On the host PC, a new scope button appears on the Manager window and a new Scope window opens.



If the Scope window is in the background, on the Manager window, click the **View Scope 1** button. The Scope window moves to the foreground.



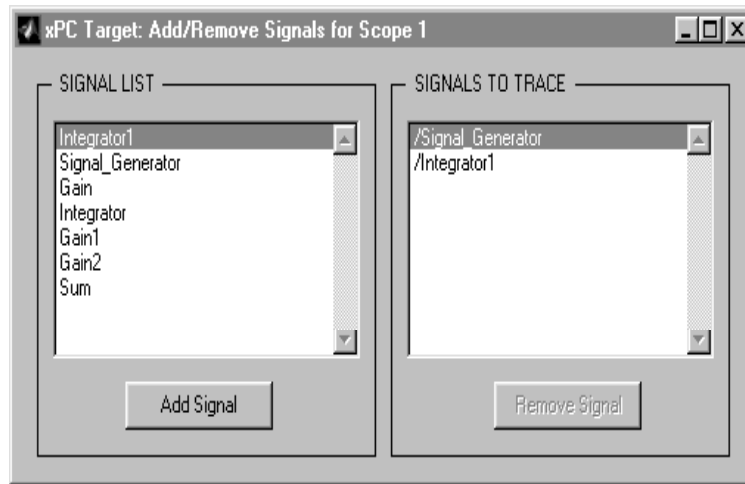
The Scope window uses most of the area for signal plotting. At the bottom, there are controls to specify the scope functions.

The target PC displays the following message.

Scope: 1, created, type is host

- 3 In the Scope window, click the **Add/Remove** button

The **Add/Remove Signals** dialog box opens. It allows you to specify which signals to trace.

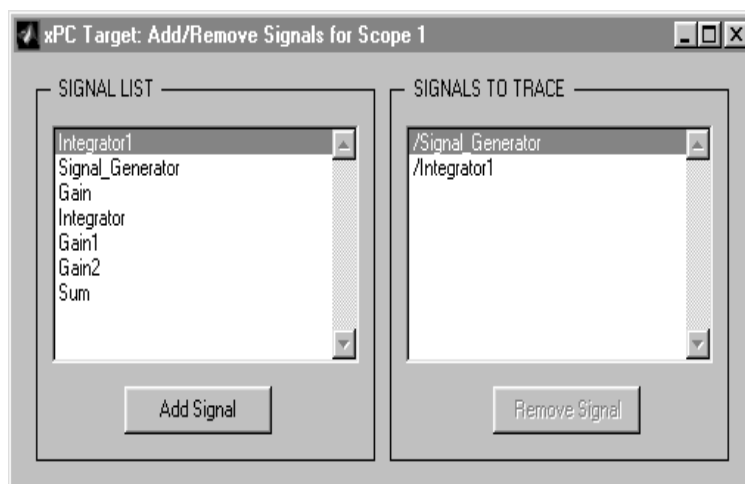


The **Signal list** box displays all of the available signals from the target application. The names of the signals correspond to the block names within the Simulink model `xpcosc.mdl`. The block name indicates the output signal from that block.

Click a block name to highlight it, and then click the **Add Signal** button to move the signal to the **Signals to trace** box on the right of the window. The **Signals to trace** box contains the signals to be traced by this scope.

- 4 From the **Signal list** box, click **Integrator 1**, and then click the **Add Signal** button. Similarly, add the **Signal Generator** signal.

Changes to the **Add/Remove Signals** dialog box are shown below. The signals to trace can be removed by clicking the block name in the **Signals to trace** box, and then clicking the **Remove Signal** button.



During the next steps, you can leave the **Add/Remove Signals** dialog box open, or close and reopen it without restrictions.

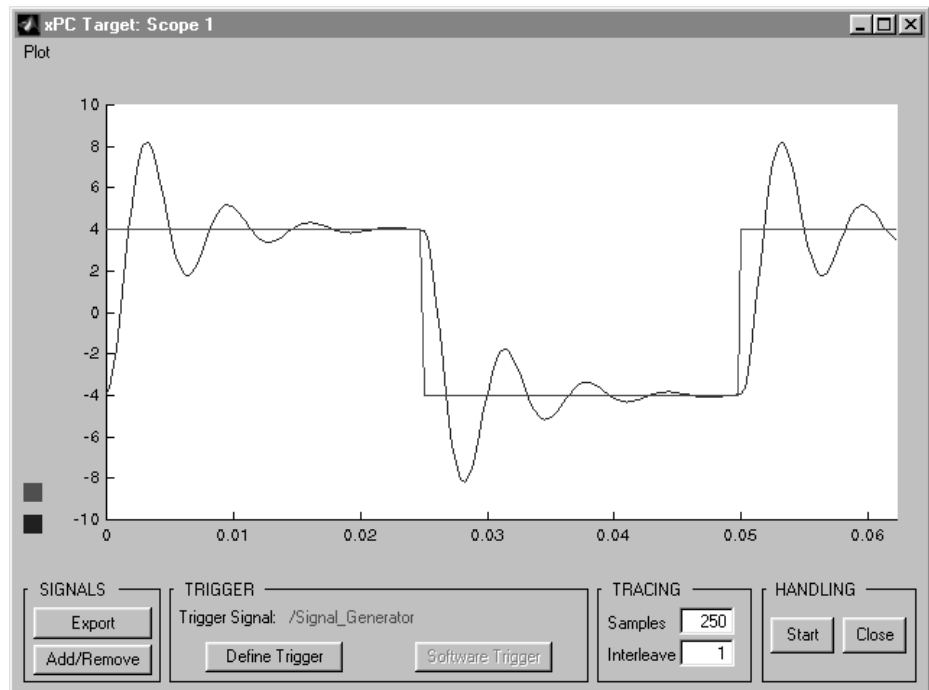
You can now start the scope, but you must also start the target application before the signals are visible in the scope window. If you use a scope, set the final time to a value high enough to ensure the target application is running during the entire signal tracing session. The final time is set by changing the target object property `tg.StopTime`.

- 5 In the Scope window, click the **Start** button. In the MATLAB window, type either

`+tg` or `tg.start` or `start(tg)`

The target application and the scope starts running. You can start the scope and the target application in any order. The target application does not have to be running to start the scope or make changes to the scope properties. While the scope is running, the **Start** button on the Scope window changes to a **Stop** button.

If a target application is running and you start a scope, the host scope window acquires one data package, and then updates the signal graph. The time to collect one data package is equal to the number of samples multiplied by the sample time.



If you are using a scope with type host, there is a delay between collecting data packages because of the communication overhead from the target PC to the host PC. If you are using a scope with type target, the scope window is updated faster than when using a scope on the host PC.

- 6 In the Scope window, click the **Stop** button.
- 7 Close the Scope Manager window by using one of the following procedures:
 - From the **File** menu, click **Close All Scopes**.
 - From the **File** menu, click **Close Scope Manager**.
- 8 A message box opens asking if you want to save the current scope state. Use one of the following procedures:
 - If you do not want to save the scope state, click **No**.
 - If you want to save the scope state, click **Yes**. The **Save Scopes** dialog box opens. Enter the name of a file, and then click **Save**.

Signal Tracing with xPC Target GUI (Target Manager)

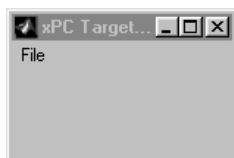
Opening a Scope window on the target PC allows you to select and view signals using a graphical user interface.

After you create, download, and start running a target application, you can view signals. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you are running the target application for that model.

- 1 In the MATLAB window, type

```
xpctgscope
```

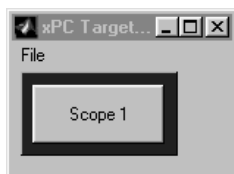
The Target Manager window opens. This window is the root-window of the Scope graphical user interface.



At this point, the window is empty because a specific scope has not been defined.

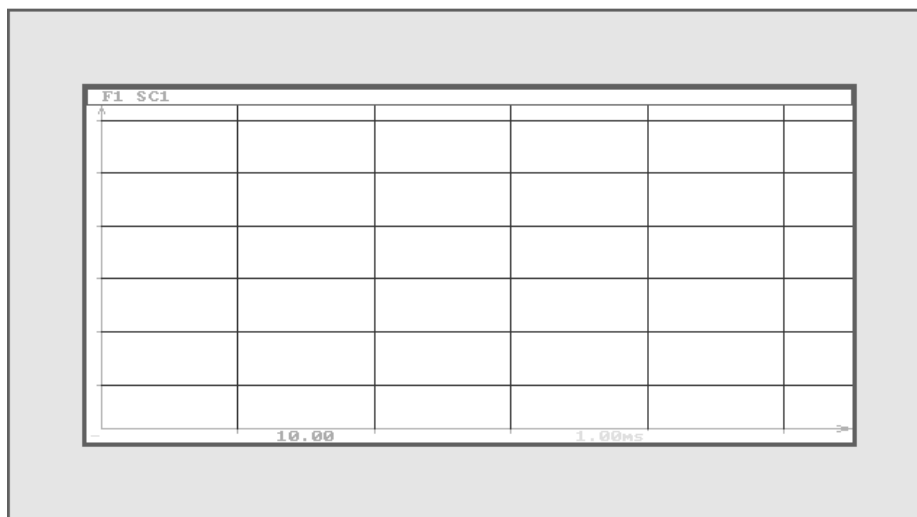
- 2 From the **File** menu, click **New Scope**.

On the host PC, and on the Target Manager window, a new scope button appears.



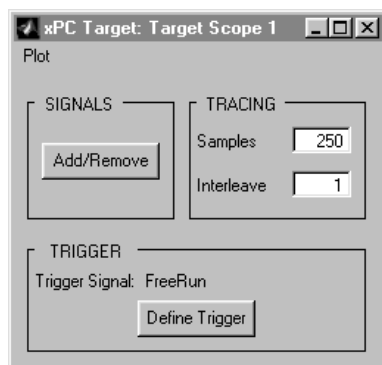
And on the target PC, a new scope window opens.

Loaded App: xpcosc Memory: 61MB Mode: RT, single Logging: t x y tet StopTime: 0.2 d SampleTime: 0.00025 AverageTET: - Execution: stopped	* xPC Target 1.0, (c) 1996-99 The MathWorks Inc. * ----- System: Host-Target Interface is RS232 (COM1/2) System: COM1 detected System: download started... System: download finished System: initializing application... System: initializing application finished Scope: 1, created, type is target
---	--



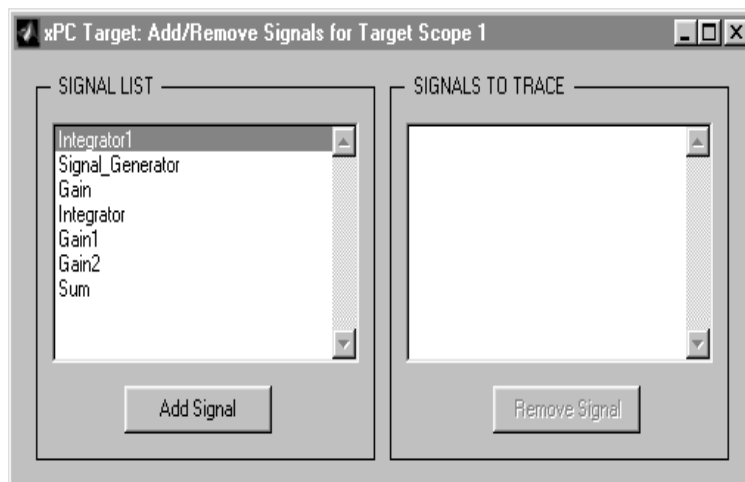
- 3 In the Target Manager window, right-click the scope button, and then click **Properties**.

The Target Scope 1 window opens.



- 4 Click the **Add/Remove** button.

The **Add/Remove Signals** dialog box opens. This dialog box allows you to specify which signals to trace.

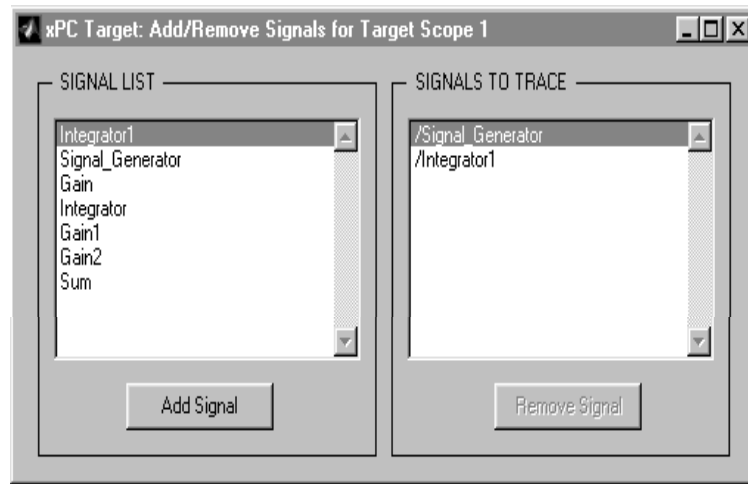


The **Signal list** box, displays all of the available signals from the target application. The names of the signals correspond to the block names within the Simulink model `xpcosc.mdl`. The block name indicates the output signal from that block.

Click a block name to highlight it, and then click the **Add Signal** button to move the signal to the **Signals to trace** box on the right of the window. The **Signals to trace** box contains the signals to be traced by this scope.

- 5 From the **Signal list** box, click **Integrator 1**, and then click the **Add Signal** button. Similarly, add the **Signal Generator** signal.

On the host PC, changes to the **Add/Remove Signals** dialog box are shown below. The signals to trace can be removed by clicking the block name in the **Signals to trace** box, and then clicking the **Remove Signal** button.



The target PC displays the following messages.

```
Scope: 1, created, type is target
Scope: 1, signal 1 added
Scope: 1, signal 0 added
```

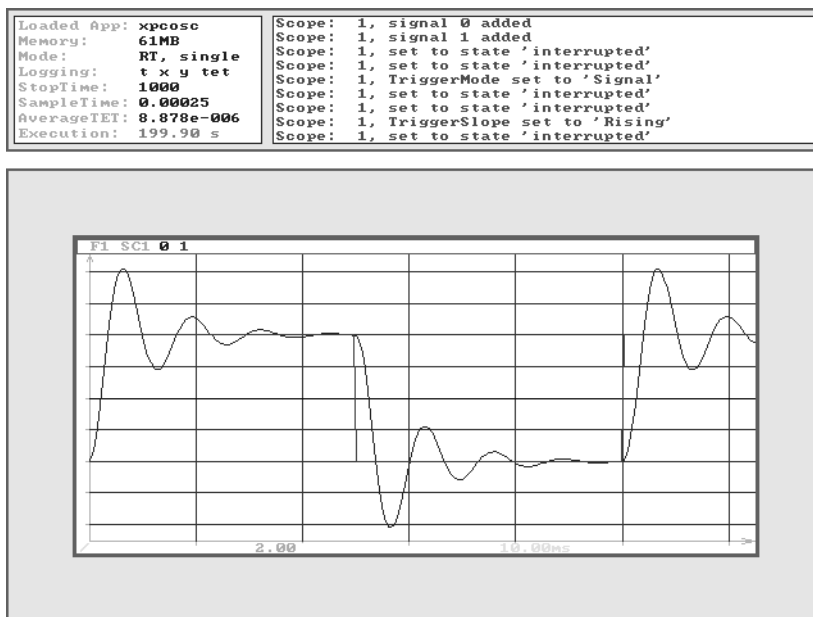
The line above the graph gives information about the target scope object. The string SC1 means that this graph corresponds to the scope object with an identifier equal to 1. The colored number 0 and number 4 are the signals added to this target scope. When you start signal tracing, the color of the traces corresponds to the color of the signal numbers.

- 6 Starting a scope on the target PC is slightly different than starting a scope on the host PC. In the Target Manager window, right-click the **Scope 1** button, and then click **Start**.

You also need to start running a target application before the signals are visible in the scope window. Type either

```
start(tg) or +tg
```


The plot window on the target PC displays the signal traces and updates at a rate equal to the time to collect one data package, as shown below.



Signal Tracing with MATLAB Commands

Creating a target scope object allows you to select and view signals using the xPC Target functions. This section describes how to signal trace using xPC Target functions instead of using the xPC Target graphical interface.

After you create and download, the target application, you can view output signals. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have build the target application for that model.

- 1 Start running your target application. Type either `+tg` or `tg.start` or `start(tg)`

The target PC displays the following message.

System: execution started (sample time: 0.0000250)

- 2 To get a list of parameters, type either

```
set(tg, 'ShowSignals', 'on') or tg.ShowSignals='on'
```

The MATLAB window displays a list of the target objects properties for the available signals. For example, the signals for the model `xpcosc.mdl` are shown below.

```
ShowSignals = On
Signals = PROP.  VALUE              BLOCK NAME
              S0      0.000000      Integrator1
              S1      0.000000      Signal Generator
              S2      0.000000      Gain
              S3      0.000000      Integrator
              S4      0.000000      Gain1
              S5      0.000000      Gain2
              S6      0.000000      Sum
```

The signal names (S0, S1 . . . S6) are properties of the target object. And the Parameter identifiers (P0, P1, . . . P6) are properties of the target object.

- 3 Create a scope to display on the target PC. For example, to create a scope with an identifier of 1 and a scope object name of `sc1`, type

```
sc1=tg.addscope('target', 1) or sc1=addscope(tg, 'target', 1)
```

- 4 List the properties of the scope object. For example, to list the properties of the scope object `sc1`, type

```
sc1
```

The MATLAB window displays a list of the scope object properties. Notice the scope properties `StartTime`, `Time`, and `Data` are not accessible with a scope of type `target`.

```
xPC Scope Object

Application      = xpcosc
ScopeId          = 1
Status           = Interrupted
Type             = Target
NumSamples       = 250
Decimation       = 1
```

TriggerMode	= FreeRun
TriggerSignal	= - 1
TriggerLevel	= 0
TriggerSlope	= Either
TriggerScope	= 1
Mode	= Redraw (Graphical)
YLimit	= Auto
Grid	= On
StartTime	= Not accessible
Data	= Not accessible
Time	= Not accessible
Signals	= no Signals defined

- 5 Add signals to the scope object. For example, to add Integrator1 and Signal Generator, type

```
tg.addsignal (sc1, [0, 1]) or addsignal (sc1, [0, 1])
```

The target PC displays the following messages.

```
Scope: 1, signal 0 added
Scope: 1, signal 1 added
```

After you add signals to a scope object, the signals are not shown on the target screen until you start the scope.

- 6 Start the scope object. For example, to start the scope sc1, type either

```
+sc1 or sc.start or start(sc1)
```

The target screen plots the signals after collecting each data package. During this time you can observe the behavior of the signals while the scope is running.

- 7 Stop the scope. Type either

```
sc1 or sc1.stop or stop(sc1)
```

The signals shown on the target PC stop updating while the target application continues running, and the target PC displays the following message.

```
Scope: 1, set to state 'interrupted'
```

Parameter Tuning

xPC Target lets you change parameters in your target application while it is running in real time.

This section includes the following topics:

- **“Parameter Tuning with MATLAB Commands”**
- **“Parameter Tuning with Simulink External Mode”**

Parameter Tuning with MATLAB Commands

You use the MATLAB functions to change block parameters. With these functions you do not need to set Simulink in external mode, and you do not need to connect Simulink with the target application.

You can download parameters to the target application while it is running or between runs. This feature lets you change parameters in your application without rebuilding the Simulink model.

After you download a target application to the target PC, you can change block parameters using MATLAB functions. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model.

- 1 In the MATLAB window, type either

```
+tg or tg.start or start(tg)
```

The target PC displays the following message.

```
System: execution started (sample time: 0.001000)
```

- 2 Display a list of parameters. Type either

```
set(tg, 'ShowParameters', 'on') or tg.ShowParameters='on'
```

and then type

```
tg
```

The MATLAB window displays a list of the target objects properties.

```
ShowParameters = On
Parameters=
```

PROP	VALUE	PARAMETER NAME	BLOCK NAME
P0	0.000000	Initial Condition	Integrator1
P1	4.000000	Amplitude	Signal Generator
P2	20.000000	Frequency	Signal Generator
P3	1000000.0	Gain	Gain
P4	0.000000	Initial Condition	Integrator
P5	400.00000	GainGain1	
P6	1000000.0	GainGain2	

The parameter names (P0, P1, . . . P6) are properties of the target object. and the signal names (S0, S1, . . .) are also properties of the target object.

- 3 Change the gain. For example, to change the Gain1 block, type either
`tg.p5=800` or `set(tg, 'p5', 800)`

As soon as you change parameters, the changed parameters in the target object are downloaded to the target application. The target PC displays the following message.

```
Param: param 5 updated
```

And the plot frame updates the signals after running the simulation with the new parameters.

- 4 Stop the target application. In the MATLAB window, type either
`-tg` or `tg.stope` or `stop(tg)`

The target application on the target PC stops running, and the target PC displays the following messages.

```
System: execution stopped
minimal TET: 0.000023 at time 1313.789000
maximal TET: 0.000034 at time 407.956000
```

Parameter Tuning with Simulink External Mode

You use Simulink external mode to connect your Simulink block diagram to your target application. The block diagram becomes a graphical user interface to your target application. You set up Simulink in external mode to establish a communication channel between your Simulink block window and your target application.

In Simulink external mode, wherever you change parameters in the Simulink block diagram, Simulink downloads those parameters to the target application while it is running. This feature lets you change parameters in your program without rebuilding the Simulink model to create a new target application.

After you download your target application to the target PC, you can connect your Simulink model to the target application. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model.

- 1 In the Simulink window, and from the **Simulation** menu, click **External**.

A check mark appears next to the menu item **External**, and Simulink external mode is activated.

- 2 In the Simulink block window, and from the **Simulation** menu, click **Connect to target**.

All of the current Simulink model parameters are downloaded to your target application. This downloading guarantees the consistency of the parameters between the host model and the target application.

The target PC displays the following message.

```
ExtM: Updating # parameters
```

- 3 From the **Simulation** menu, click **Start real-time code** or in the MATLAB window, type either

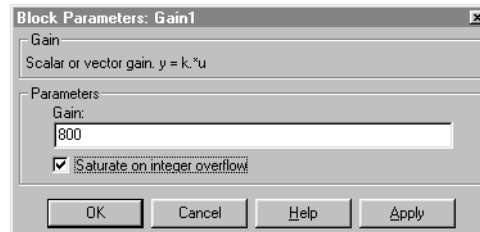
```
+tg or tg.start or start(tg)
```

The target application begins running on the target PC, and the target PC displays the following message.

```
System: execution started (sample time: 0.000250)
```

- 4 From the Simulation block diagram, click the block labeled **Gain1**.

The **Block Parameters: Gain1** parameter dialog box opens.



- 5 In the **Gain** text box, enter 800, and click **OK**.

As soon as you change a model parameter and click **OK** or the **Apply** button on the **Block Parameters: Gain1** dialog box, all of the changed parameters in the model are downloaded to the target application, as shown below.

Loaded App: xpcosc	Scope: 1, set to state 'interrupted'
Memory: 61MB	Scope: 1, TriggerSlope set to 'Rising'
Mode: RT, single	Scope: 1, set to state 'interrupted'
Logging: t x y tet	System: execution stopped at 553.624250
StopTime: 1000	minimal TET: 0.000008 at time 0.007500
SampleTime: 0.00025	maximal TET: 0.000019 at time 229.064750
AverageTET: 8.824e-006	System: execution started (sample time: 0.000250)
Execution: 54.80 s	ExtM: updating 7 parameters
	ExtM: updating parameter



- 6 From the **Simulation** menu, click **Disconnect from Target**.

The Simulink model is disconnected from the target application. Now, if you change a block parameter in the Simulink model, there is no effect on the target application. Connecting and disconnecting Simulink works regardless of whether the target application is running or not.

- 7 From the **Simulation** menu, click **Stop real-time code**, or in the MATLAB command window, type either

```
stop(tg) or -tg
```

The target application on the target PC stops running, and the target PC displays the following messages.

```
System: execution stopped
minimal TET: 0.000023 at time 1313.789000
maximal TET: 0.000034 at time 407.956000
```


Advanced Procedures

I/O Driver Blocks	4-3
xPC Target I/O Driver Blocks	4-3
Adding I/O Blocks with the xPC Target Library	4-4
Adding I/O Blocks with the Simulink Library Browser	4-7
Defining I/O Block Parameters	4-10
 xPC Target Scope Blocks	 4-13
xPC Target Scope Blocks	4-13
Adding xPC Target Scope Blocks	4-14
Defining xPC Target Scope Block Parameters	4-16
 Target PC Command Line Interface	 4-19
Using Methods and Properties on the Target PC	4-19
Target Object Methods	4-20
Target Object Properties	4-20
Scope Object Methods	4-21
Scope Object Properties	4-22
Using Variables on the Target PC	4-24
Variable Commands	4-24
 Web Interface	 4-25
Connecting the Web Interface	4-25
Using the Main Page	4-26
Changing WWW Properties	4-28
Viewing Signals with the Web Browser	4-28
Using Scopes with the Web Browser	4-29
Viewing and Changing Parameters with the Web Interface	4-30
Changing Access Levels to the Web Browser	4-31

After learning the basic procedures for creating and running a target application, signal acquisition and parameter tuning, you can try some of the special and advanced procedures with xPC Target.

This chapter includes the following sections:

- **“I/O Driver Blocks”** - Adding I/O driver blocks to your Simulink model connects your model to sensors and actuators.
- **“xPC Target Scope Blocks”** - Adding xPC Target scopes blocks to your Simulink model eliminates the need to create and define scopes after the build process.
- **“Web Interface”** - Connect to the target application from any computer connected to the network.
- **“Target PC Command Line Interface”** - Enter commands on the target PC for stand alone applications not connected to the host PC.

I/O Driver Blocks

You add I/O driver blocks to your Simulink model to connect your model to physical I/O boards. These I/O boards then connect to the sensors and actuators in the physical system.

This section includes the following topics:

- **“xPC Target I/O Driver Blocks”**
- **“Adding I/O Blocks with the xPC Target Library”**
- **“Adding I/O Blocks with the Simulink Library Browser”**
- **“Defining I/O Block Parameters”**

xPC Target I/O Driver Blocks

A driver block does not represent an entire board, but an I/O section supported by a board. Therefore, the xPC Target library may have more than one block for each physical board. I/O driver blocks are written as C-code S-functions (noninlined S-functions). We include the source code for the C-code S-functions with xPC Target.

xPC Target supports PCI and ISA busses. If the bus type is not indicated in the driver block number, you can determine the bus type of a driver block by checking the blocks parameter dialog box. The last parameter is either a PCI slot for PCI boards, or a Base Address for ISA boards.

Adding I/O Blocks with the xPC Target Library

xPC Target contains an I/O driver library with Simulink blocks. You can drag-and-drop these blocks from the library to your Simulink model. Alternately, you can access the I/O driver library with the Simulink Library Browser. See “Adding I/O Blocks with the Simulink Library Browser” on page 4-7.

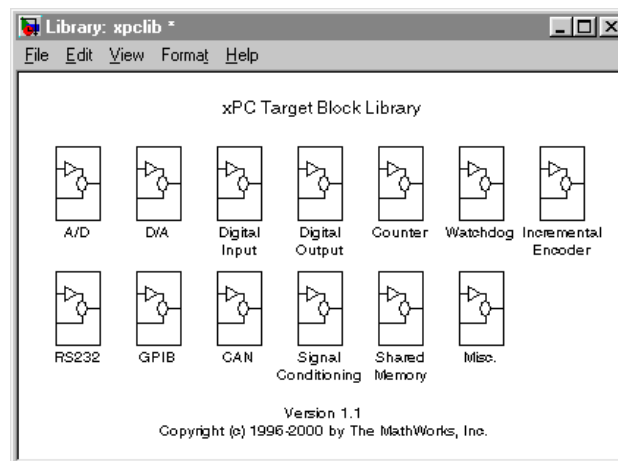
The highest hierarchy level in the library is grouped by I/O function. I/O functions include A/D, D/A, Digital In, Digital Out, Counter, Watchdog, Incremental Encoder, RS232, CAN, GPIB, and shared memory. The second level is grouped by board manufacturer. The manufacturer groups within this second level contain the specific boards.

This procedure uses the Simulink model `xpcosc.mdl` as an example of how to connect an I/O block.

- 1 In the MATLAB window, type

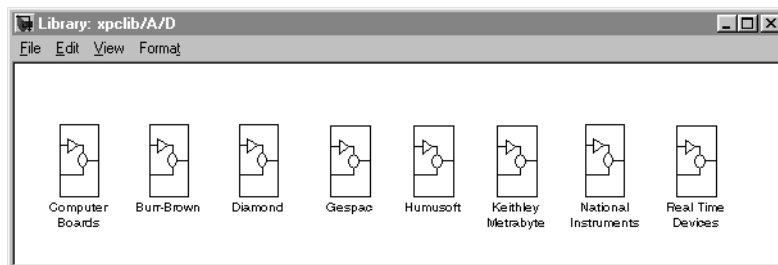
```
xpclib
```

The Library: `xpclib` window opens.



- 2 Open a function group. For example, to open the A/D group, double-click the **A/D** block.

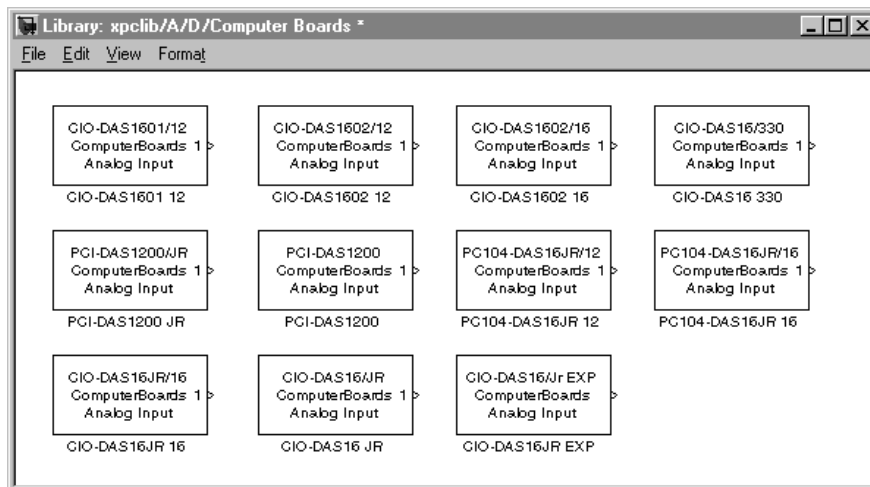
The manufacturer level opens.



Within each manufacturer group are the blocks for a single function.

- 3 Open a manufacturer group. For example, to open the A/D driver blocks from ComputerBoards, double-click the group marked **ComputerBoards**.

The window with the A/D driver blocks for ComputerBoards opens.



- 4 In the Simulink window, type
xpcosc

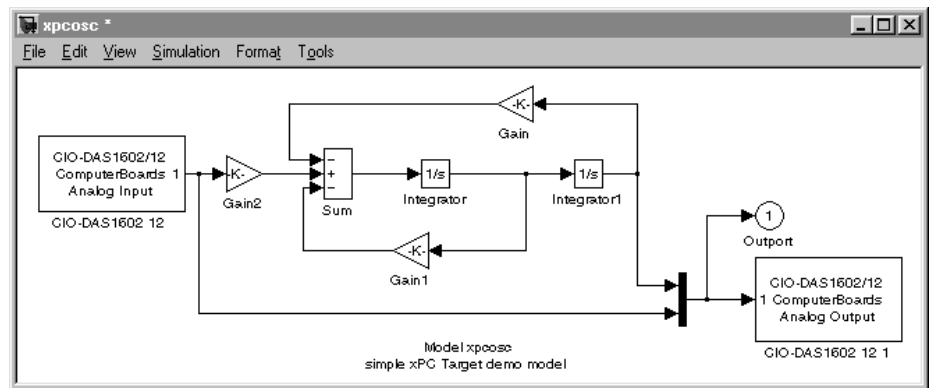
The Simulink block diagram opens for the model xpcosc.mdl.

- 5 From the block library, click-and-drag the name of an A/D board to the Simulink block diagram. Likewise, click-and-drag the name of a D/A board to your model.

Simulink adds the new I/O blocks to your model.

- 6 Remove the signal generator block and add the analog input block in its place. Remove the scope block and add the analog output block in its place.

The demo model `xpcosc` should look like the figure shown below.



Your next task is to define the I/O block parameters. See “Defining I/O Block Parameters” on page 4-10.

Adding I/O Blocks with the Simulink Library Browser

You can access the xPC Target driver blocks using the Simulink Library Browser. You drag-and-drop these blocks from the library to your Simulink model. Alternately, you can access driver blocks using the xPC Target I/O driver library. See “Adding I/O Blocks with the xPC Target Library” on page 4-4.

The highest hierarchy level in the library is grouped by I/O function. I/O functions include A/D, D/A, Digital In, Digital Out, Counter, Watchdog, Incremental Encoder, RS232, CAN, GPIB, and shared memory. The second level is grouped by board manufacturer. The manufacturer groups within this second level contain the specific boards.

This procedure uses the Simulink model `xpcosc.mdl` as an example of how to connect an I/O block.

- 1 In the MATLAB window type

```
xpcosc
```

The Simulink block diagram opens for the model `xpcosc.mdl`.

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

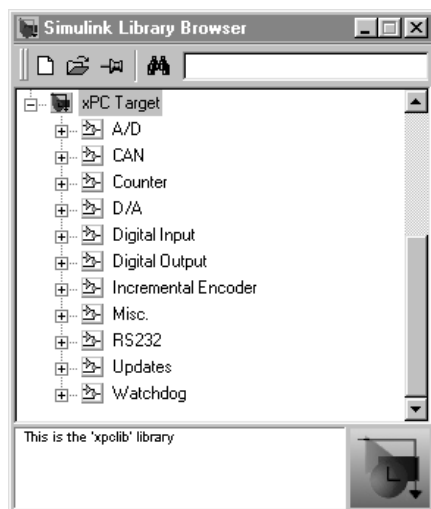
The Simulink Library Browser window opens. Alternately, you can open the Simulink Library Browser by typing `simulink` in the MATLAB command window.



You can access the xPC Target I/O library by right-clicking **xPC Target**, and then clicking **Open the xPC Target Library**.

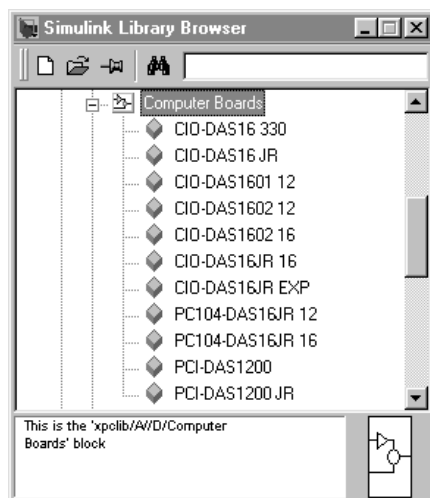
3 Double-click **xPC Target**.

A list of I/O functions opens.



- 4 Open a function group. For example, to open the A/D group for ComputerBoards, double-click **A/D**, and then double-click **ComputerBoards**.

A list with the A/D driver blocks for ComputerBoards opens.

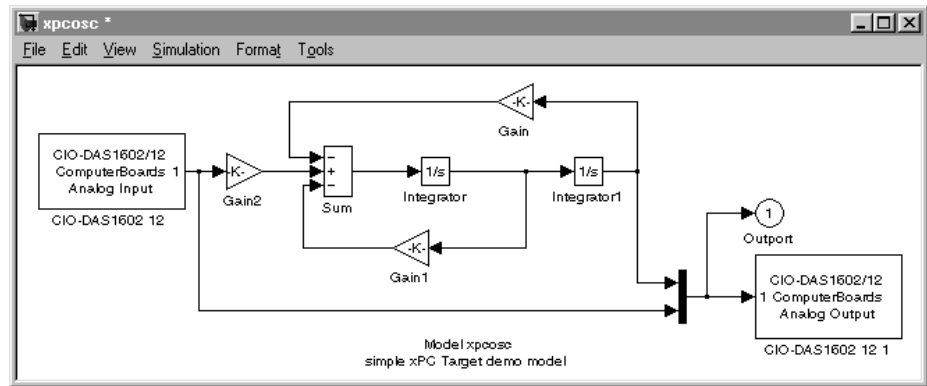


- 5 From the block library, click-and-drag the name of an A/D board to the Simulink block diagram. Likewise, click-and-drag the name of a D/A board to your model

Simulink adds the new I/O blocks to your model.

- 6 Remove the signal generator block and add the analog input block in its place. Remove the scope block and add the analog output block in its place.

The model xpcosc should look like the figure shown below.



Your next task is to define the I/O block parameters. See “Defining I/O Block Parameters”.

Defining I/O Block Parameters

The I/O block parameters define values for your physical I/O boards. For example, I/O block parameters include channel numbers for multichannel boards, input and output voltage ranges, and sample time.

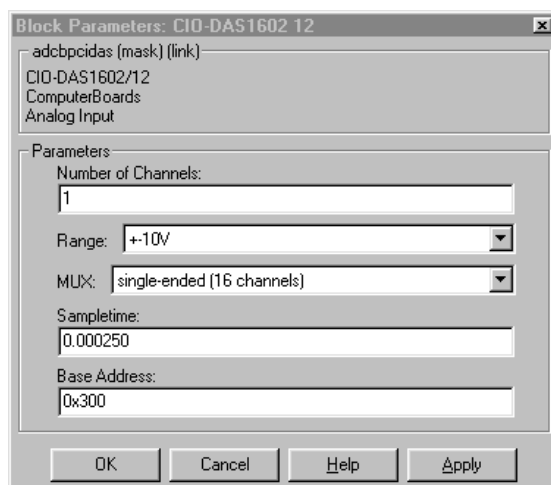
This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have added an analog input and an analog output block to your model. To add an I/O block, see either “Adding I/O Blocks with the xPC Target Library” on page 4-4 or “Adding I/O Blocks with the Simulink Library Browser” on page 4-7.

- 1 In the Simulink window, double-click the input block labeled **Analog Input**.

The dialog box for the A/D converter opens.

- 2 Fill in the dialog box. For example, for a single channel enter 1 in the Number of Channels box, choose ± 10 V for the input range, and choose single-ended (16 channels) for the MUX-switch position. Enter the same sample time you entered for the step size in the Simulation Parameters dialog box. Enter the base address for this ISA-bus board.

The **Block Parameters** dialog box should look similar to the figure shown below.

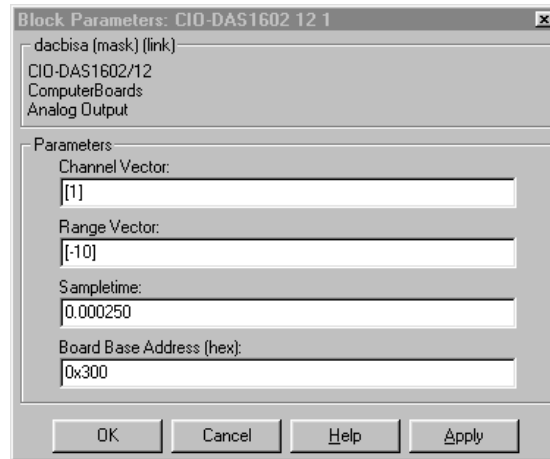


- 3 In the Simulink window, double-click the output block labeled **Analog Output**.

The dialog box for the D/A converter opens.

- 4 Fill in the dialog box. For example, for one channel enter [1] in the Channel Vector box, for an output level of ± 10 V enter the code [-10] in the Range Vector box. Enter the same sample time you entered for the step size in the Simulation Parameters dialog box. Enter the base address for this ISA-bus board.

The **Block Parameters** dialog box should look similar to the figure shown below.



If you change the sample time by changing the target object property `SampleTime`, the step size you entered in the **Simulation Parameters** dialog box, as well as the sample times you entered in both of the I/O blocks are set to the new value.

Your next task is to build and run the target application, see “Creating the Target Application” on page 3-7.

xPC Target Scope Blocks

Usually scope objects are defined using xPC Target functions or the graphical interface after downloading your target application. An alternative is for you to add special xPC Target scope blocks to your Simulink model. These blocks should not be confused with standard Simulink scope blocks. The xPC Target scope blocks have unique capabilities when used with xPC Target.

This section includes the following topics:

- “**xPC Target Scope Blocks**”
- “**Adding xPC Target Scope Blocks**”
- “**Defining xPC Target Scope Block Parameters**”

xPC Target Scope Blocks

An xPC Target scope block is added to your model the same way you add any Simulink block. After adding an xPC Target scope block, you define the properties of the scope object and the signals you want to display. When the target application is downloaded to the target PC, the scope is automatically created on the target PC. No additional definitions are necessary.

Note The scope object created on the host PC is not assigned to a MATLAB variable. To assign the scope object, use the target object function `getscope`. Also, if you use the function `remscope` to remove a scope created during the build and download process, and then you restart the target application, the scope is recreated.

Alternative methods for creating xPC Target scopes - For information on using xPC Target functions to create scopes, see “Signal Tracing with MATLAB Commands” on page 3-35. For information on using the xPC Target graphical interface to create scopes, see “Signal Logging with xPC Target Graphical Interface” on page 3-20.

Adding xPC Target Scope Blocks

Adding xPC Target scope blocks to your Simulink model saves you the time to define and select signals after you download the target application to the target PC, and the information is saved with your model.

You can drag an xPC Target scope block into any Simulink model, and the input to the scope can be connected to any block output. If you want to trace more than one signal, add a multiplexer block to your model, and connect the scope block to the multiplexer output.

The following procedure uses the Simulink model `xpcosc.mdl` as an example to show how to connect an xPC Target scope block to your model.

- 1 In the MATLAB window type

`xpcosc`

The Simulink block diagram opens for the model `xpcosc.mdl`.

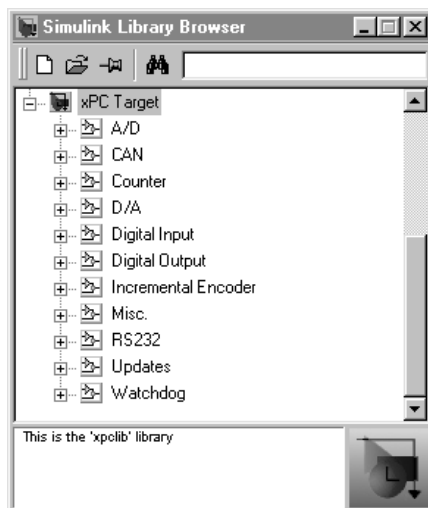
- 2 In the Simulink window, and from the **View** menu, click **Show Library Browser**.

The Simulink Library Browser window opens.



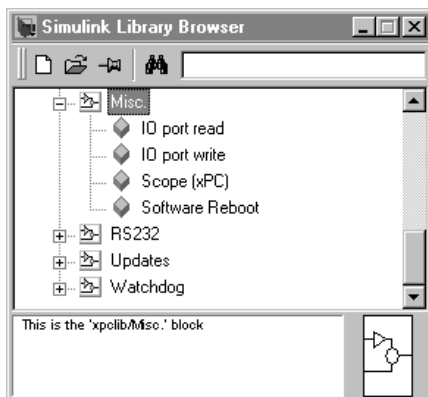
3 Double-click **xPC Target**.

A list of I/O functions opens.



4 Double-click **Misc.**

A list of miscellaneous group blocks opens.

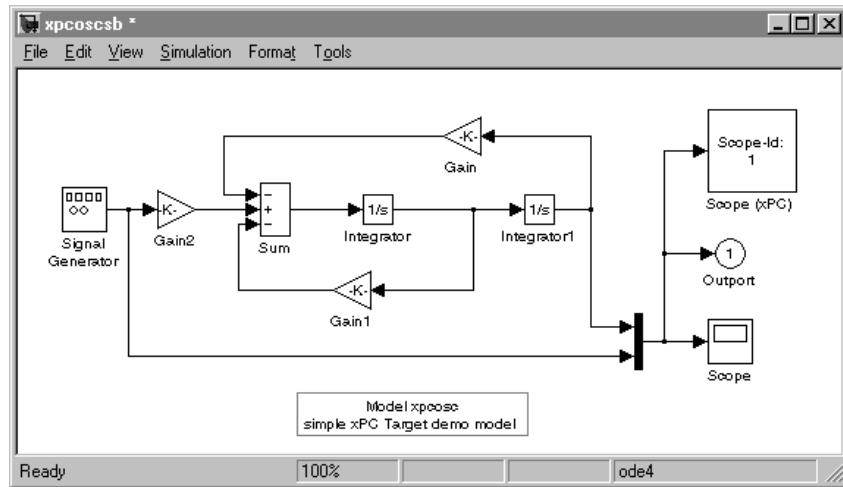


- 5 Click-and-drag **Scope (xPC)** to your Simulink block diagram.

Simulink adds a new scope block to your model with a scope identifier of 1.

- 6 Connect the xPC Target scope block with the Simulink scope block.

The model `xpcosc` should look like the figure shown below.



Your next task is to define the xPC Target scope block parameters. See “Defining xPC Target Scope Block Parameters”.

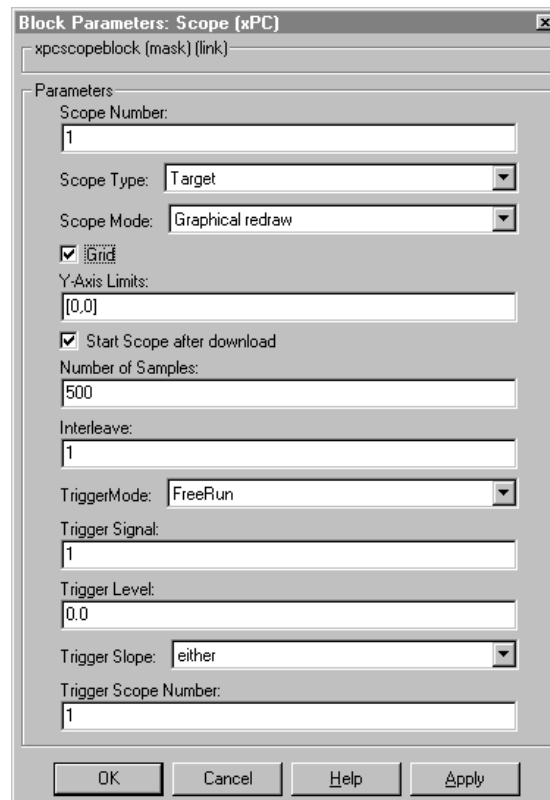
Defining xPC Target Scope Block Parameters

Scope block parameters define the signals to trace on the scope, trigger modes, and the axis range.

This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have added an xPC Target scope block to your model. To add a scope block, see “Adding xPC Target Scope Blocks” on page 4-14.

- 1 Double-click the scope block labeled **Scope (xPC)**.

The **Block Parameters** dialog box for the scope block opens.



The image shows a dialog box titled "Block Parameters: Scope (xPC)". It contains the following fields and controls:

- Scope Number: 1
- Scope Type: Target
- Scope Mode: Graphical redraw
- ☒ Grid
- Y-Axis Limits: [0,0]
- ☒ Start Scope after download
- Number of Samples: 500
- Interleave: 1
- TriggerMode: FreeRun
- Trigger Signal: 1
- Trigger Level: 0.0
- Trigger Slope: either
- Trigger Scope Number: 1

At the bottom are buttons for OK, Cancel, Help, and Apply.

- 2 In the **Scope Number** box, enter a unique number to identify the scope. This number is used to identify the xPC Target scope block and the scope screen on the host or target computers.
- 3 From the **Scope Type** list, choose either **Host** or **Target**.
- 4 From the **Scope Mode** list, choose either **Numerical**, **Graphical (redraw)**, **Graphical (sliding)**, or **Graphical (rolling)**.

If you selected **Host** for the Scope Type, then you can only choose either **Numerical** or **Graphical (redraw)** for the Scope Mode.

- 5 Select the **Grid** check box to display grid lines on the scope.

- 6 In the **Y-axis Limits** box, enter a row vector with two elements where the first element is the lower limit of the y-axis and the second element is the upper limit. If you enter 0 for both elements, then the scaling is set to auto.
- 7 Select the **Start Scope after download** check box, to start a scope when the target application is downloaded. With a target scope, the scope window open automatically. With a host scope, you need to open the window by typing `xpcscope`.
- 8 In the **Number of Samples** box, enter the number of values acquired in a data package before redrawing the graph. In the **Interleave** box, enter a value to collect data at each sample time (1) or to collect data at less than every sample time (2 or greater).
- 9 From the **Target Mode** list, choose either **FreeRun**, **Software Trigger**, **Signal Trigger**, or **Scope Trigger**.

If you selected **Signal Trigger**, then in the **Trigger Signal** box, enter the index of a signal. In the **Trigger Level** text box, enter a value for the signal to cross before triggering. And from the **Trigger Slope** list choose **either**, **rising**, or **falling**.

If you choose **Scope Trigger**, then in the **Trigger Scope Number** box, enter the block number of a scope block. If you use this feature, you must also add a second scope block to your Simulink model.

- 10 Click **OK**.

Your next task is to build and run the target application. As soon as the target application is built and downloaded, a scope and scope object are automatically created. However, the scope object is not assigned to a MATLAB variable.

Target PC Command Line Interface

You can interact with the xPC Target environment through the target PC command window. This interface is useful with stand-alone applications that are not connected to the host PC.

This section includes the following topics:

- “Using Methods and Properties on the Target PC”
- “Target Object Methods”
- “Target Object Properties”
- “Scope Object Methods”
- “Scope Object Properties”
- “Using Variables on the Target PC”
- “Variable Commands”

Using Methods and Properties on the Target PC

xPC Target uses an object oriented environment on the host PC with methods and properties. While the target PC does not use the same objects, many of the methods on the host PC have equivalent target PC commands. The target PC commands are case sensitive, but the arguments are not case sensitive.

After you have created and downloaded a target application to the target PC, you can use the target PC commands to run and test your application.

- 1 On the target PC, press C or move the mouse over the command window.

The target PC command window is activated, and a command line opens. If the command window is already activated, you do not have to press C. In this case, pressing C is taken as the first letter in a command.

- 2 In the **Cmd** box, type a target PC command. For example, to start your target application, type

`start <enter>`

Once the command window is active, you do not have to reactivate it before typing the next command. For a list of target PC commands, see “Target Object Methods” on page 4-20, “Target Object Properties” on page 4-20, “Scope Object Methods” on page 4-21, and “Scope Object Properties” on page 4-22.

Target Object Methods

When using the target PC command line interface, target object methods are limited to starting and stopping the target application.

The following table lists the syntax for the target commands that you can use on the target PC. The MATLAB equivalent syntax is shown in the right column, and the target object name `tg` is used as an example for the MATLAB methods.

Target PC	MATLAB
<code>start</code>	<code>tg.start</code> or <code>+tg</code>
<code>stop</code>	<code>tg.stop</code> or <code>-tg</code>
<code>reboot</code>	<code>tg.reboot</code>

Target Object Properties

When using the target PC command line interface, target object properties are limited to parameters, signals, `stoptime`, and `sampletime`. Notice the difference between a parameter index (0, 1, . . .) and a parameter name (P0, P1, . . .).

The following table lists the syntax for the target commands that you can use on the target PC. The MATLAB equivalent syntax is shown in the right column, and the target object name `tg` is used as an example for the MATLAB methods.

Target PC	MATLAB
<code>setpar parameter_index = number</code>	<code>set(tg, 'parameter_name', number)</code>
<code>getpar parameter_index</code>	<code>get(tg, 'parameter_name')</code>
<code>stoptime = number</code>	<code>tg.stoptime = number</code>
<code>sampletime = number</code>	<code>tg.sampletime = number</code> <code>set(tg, 'SampleTime', number)</code>

Target PC	MATLAB
<i>parameter_name</i> (P0, P1, . . .) <i>parameter_name</i> = <i>number</i>	tg. <i>parameter_name</i> tg. <i>parameter_name</i> = <i>number</i>
<i>signal_name</i> (S0, S1, . . .)	tg. S#

Scope Object Methods

When using the target PC command line interface, you use scope object methods to start a scope, and add signal traces. Notice the methods `addscope` and `remscope` are target object methods on the host PC, and notice the difference between a signal index (0, 1, . . .) and a signal name (S0, S1, . . .)

The following table lists the syntax for the target commands that you can use on the target PC. The MATLAB equivalent syntax is shown in the right column. The target object name `tg` and the scope object name `sc` is used as an example for the MATLAB methods.

Target PC	MATLAB
<code>addscope scope_index</code> <code>addscope</code>	tg. <code>addscope(scope_index)</code> tg. <code>addscope</code>
<code>remscope scope_index</code> <code>remscope 'all'</code>	tg. <code>remscope(scope_index)</code> tg. <code>remscope</code>
<code>startscope scope_index</code>	sc. <code>start</code> or <code>+sc</code>
<code>stopscope scope_index</code>	sc. <code>stop</code> or <code>-sc</code>
<code>addsignal scope_index =</code> <code>signal_index1, signal_index2,</code> <code>. . .</code>	sc. <code>addsignal(signal_index_vector)</code>
<code>remsignal scope_index =</code> <code>signal_index1, signal_index2,</code> <code>. . .</code>	sc. <code>remsignal(signal_index_vector)</code>

Target PC	MATLAB
<code>viewmode scope_index</code> or left click the scope window	
<code>viewmode all</code> or right click any scope window	
Press function key for scope, and then press V to toggle viewmode	
<code>ylimt scope_index</code> <code>ylimt scope_index = auto</code> <code>ylimt scope_index = num1, num2</code>	
<code>grid on</code> <code>grid off</code>	

Scope Object Properties

When using the target PC command line interface, scope object properties are limited to those shown in the following table. Notice the difference between a scope index (0, 1, . . .) and the MATLAB variable name for the scope object on the host PC. The scope index is indicated in the top left corner of a scope window (SC0, SC1, . . .).

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

The following table lists the syntax for the target commands that you can use on the target PC. The MATLAB equivalent syntax is shown in the right column, and the scope object name `sc` is used as an example for the MATLAB methods

Target PC	MATLAB
<code>numsamples scope_index = number</code>	<code>sc.NumSamples = number</code>
<code>decimation scope_index = number</code>	<code>sc.Decimation = number</code>

Target PC	MATLAB
scopemode <i>scope_index</i> = 0 or numerical, 1 or redraw, 2 or sliding, 3 or rolling	sc.Mode = 'numerical', 'redraw', 'sliding', 'rolling'
triggermode <i>scope_index</i> = 0, freerun, 1 software, 2, signal, 3, scope	sc.TriggerMode = 'freerun', 'software', 'signal', 'scope'
numprepostsamples <i>scope_index</i> = <i>number</i>	sc.NumPrePostSamples = <i>number</i>
triggersignal <i>scope_index</i> = <i>signal_index</i>	sc.TriggerSignal = <i>signal_index</i>
triggerlever <i>scope_index</i> = <i>number</i>	sc.TriggerLevel = <i>number</i>
triggerslope <i>scope_index</i> = 0, either, 1, rising, 2, falling	sc.TriggerSlope = 'Either', 'Rising', 'Falling'
triggerscope <i>scope_index2</i> = <i>scope_index1</i>	sc.TriggerScope = <i>scope_index1</i>
Press function key for scope, and then press S or move mouse into the socpe window	sc.trigger

Using Variables on the Target PC

Use variables to tag unfamiliar commands, parameter indices, and signal indexes with more descriptive names.

After you have created and downloaded a target application to the target PC, you can create target PC variables.

- 1 On the target PC, press C.

The target PC command window is activated, and a command line opens.

- 2 In the **Cmd** box, type a variable command. For example, if you have a parameter that controls a motor, you could create the variables `on` and `off` by typing

```
setvar on = p7= 1
setvar off = p7=0
```

- 3 Type a variable name. For example, to turn the motor on, type
`on`

The parameter P7 is changed to 1, and the motor turns on.

Variable Commands

The following table lists the syntax for the target commands that you can use on the target PC. The MATLAB equivalent syntax is shown in the right column.

Target PC	MATLAB
<code>setvar variable_name = target command</code>	
<code>getvar variable_name</code>	
<code>del var variable_name</code>	
<code>del all var</code>	

Web Interface

xPC Target has a Web server build in the TCP/IP mode to the kernel that allows you to interact your target application using a Web browser. If the target PC is connected to a network, you can use a Web browser to interact with the target application from any computer connected to a network

Currently this feature is limited to Microsoft Internet Explorer (version 4.0 or later) and Netscape Navigator (version 4.5 or later) are the supported browsers.

This section includes the following sections:

- **“Connecting the Web Interface”**
- **“Using the Main Page”**
- **“Changing WWW Properties”**
- **“Viewing Signals with the Web Browser”**
- **“Using Scopes with the Web Browser”**
- **“Viewing and Changing Parameters with the Web Interface”**
- **“Changing Access Levels to the Web Browser”**

Connecting the Web Interface

The TCP/IP stack on the xPC Target kernel supports only one simultaneous connection, since its main objective is real-time applications. This connection is shared between MATLAB and the Web browser. This also means that only one browser or MATLAB is able to connect at one time.

Before you connect your Web browser, a target application must be loaded onto the target PC. Whether the target application is running or not is irrelevant, but it must be loaded. Also, JavaScript and StyleSheets must be turned on.

- 1 In the MATLAB window, type

```
xpcwwenable
```

MATLAB is disconnected from the target PC, and the connection is reset for connecting to another client. If you do not use this command, your Web browser may not be able to connect to the target PC.

- 2 Open a Web browser. In the address box, enter the IP address and port number you entered in the xPC Target Setup window. For example, if the target computer IP address is 192.168.0.1 and the port is 22222, type

http://192.168.0.1:22222/

The browser loads the xPC Target Web interface frame and pages.

Using the Main Page

The Main page is divided into four parts, one below the other. The four parts are: System Status, xPC Target Properties, Navigation, and WWW Properties.

After you connect a Web browser to the target PC you can use the Main page to control the target application.

- 1 In the left frame, click the **Refresh** button.


System status information in the top cell is uploaded from the target PC. If the right frame is either the Signals List page or the Screen Shot page, updating the left frame also updates the right frame.

System Status	
Application	xpcosc
Mode	Real-Time Single-Tasking
Status	Stopped
CPUOverload	none
ExecTime	0.0
SessionTime	13305
StopTime	Inf
SampleTime	0.00025
AvgTET	2.70114e-005

- 2 Click the **Start Execution** button.

The target application begins running on the target PC, the Status line is changed from Stopped to Running, and the **Start Execution** button text changes to **Stop Execution**.

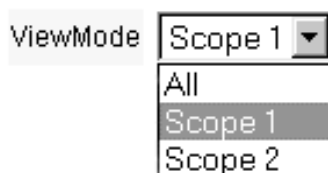
- 3 Update the execution time and average task execution time (TET). Click the **Refresh** button. To stop the target application, click the **Stop Execution** button.
- 4 Enter new values in the **Stop Time** and **Sample Time** boxes, and then click the **Apply** button. You can enter - 1 or Inf in the Stop Time box for an infinite stop time.



The screenshot shows a configuration panel with two input fields. The first field is labeled 'SampleTime' and contains the value '0.00025'. The second field is labeled 'StopTime' and contains the value '1000'. Below these fields are two buttons: 'Apply' and 'Reset'.

The new property values are downloaded to the target application. Notice, the sample time box is visible only when the target application is stopped. You cannot change the sample time while a target application is running.

- 5 Select scopes to view on the target PC. From the **ViewMode** list choose one or all of the scopes to view.



The screenshot shows a dropdown menu labeled 'ViewMode'. The menu is open, displaying three options: 'All', 'Scope 1', and 'Scope 2'. 'Scope 1' is currently selected and highlighted.

Note the **ViewMode** button is visible only if you add two or more scopes to the target PC.

Changing WWW Properties

The WWW Properties cell in the left frame contains fields that affect the display on the Web interface itself, and not the application. There are two fields: maximum signal width to display, and refresh interval.

- 1 In the **Maximum Signal Width** enter - 1, Inf (all signals), 1 (show only scalar signals), 2 (show scalar and vector signals less than or equal to 2 wide), or n (show signals with a width less than n)

Signals with a width greater than the value you enter, are not displayed on the Signals page.

- 2 In the **Refresh Interval** box, enter a value greater than 10. For example, enter 20.

The signal page updates automatically every 20 seconds. Entering - 1 or Inf does not automatically refresh the page.

Sometimes, both of the frames try to update simultaneously, or the auto refresh starts before the previous load has finished. This problem may happen with slow network connections. In this case, increase the refresh interval or manually refresh the browser (Set the Refresh Interval = Inf).

This may also happen when you are trying to update a parameter or property at the same time as the page is automatically refreshing.

Sometimes, when a race condition occurs, the browser becomes confused about the format, and you may have to refresh it. This should not happen too often.

Viewing Signals with the Web Browser

The Signal page is a list of the signals in your model.

After you connect a Web browser to the target PC you can use the Signals page to view signal data.

- 1 In the left frame, click the **Signals** button.

The Signals page is loaded in the right frame with a list of signals and the current values.

- 2 On the Signals page in the right frame, click the **Refresh** button.

The Signals page is updated with the current values. Vector/matrix signals are expanded and indexed in the same column-major format that MATLAB uses. This may be affected by the Maximum Signal Width value you enter in the left frame.

- 3 In the left frame, click the **Screen Shot** button.

The Screen Shot page is loaded and a copy of the current target PC screen is displayed. The screen shot uses the Portable Network Graphics file format PNG.

Using Scopes with the Web Browser

The Web browser interface allows you to visualize data using an interactive graphical interface.

After you connect a Web browser to the target PC you can use the Scopes page to add, remove and control scopes on the target PC.

- 1 In the left frame, click the **Scopes** button.

The Scopes List page is loaded into the right frame.

- 2 Click the **Add Scope** button.

A scope of type target is created and displayed on the target PC. The scopes page displays a list of all the scopes present. The capability exists to add a new scope, remove existing scopes, and control all aspects of a scope from this page.

Note If any host scopes exist, they will be visible and controllable from this page. They may even be removed. However, there is no capability to add any scopes of type host from the Scope page.

- 3 Click the **Edit** button.

The scope editing page opens. From this page, you can edit the properties of any scope, and control the scope.

- 4 Click the **Add Signals** button.

The browser displays an **Add New Signals** list.

- 5 Select the check boxes next to the signal names, and then click the **Apply** button.

A **Remove Existing Signals** list is added above the **Add New Signals** list.

You do not have to stop a scope to make changes. If stopping the scope is necessary, the Web interface stops it automatically and then restarts it if necessary when the changes are made. It will not restart the scope if the state was originally stopped.

Viewing and Changing Parameters with the Web Interface

The parameters page displays a list of all the tunable parameters in your model. Row and column indices for vector/matrix parameters are also shown.

After you connect a Web browser to the target PC you can use the Parameters page to change parameters in your target application while it is running in real time.

- 1 In the left frame, click the **Parameters** button.

The Parameter List page is loaded into the right frame.

If the parameter is a scalar parameter, the present parameter value is shown in a box that you can edit.

If the parameter is a vector/matrix, there is a button which takes you to another page which displays the vector/matrix (in the correct shape) and enables you to edit the parameter

- 2 Enter a new parameter value into one or more of the parameter boxes, and then click the **Apply** button.

The new parameter values are uploaded to the target application.

Changing Access Levels to the Web Browser

The Web browser interface allows you to set access levels to the target application. The different levels limit access to the target application. The highest level, 0, is the default level and allows full access. The lowest level 4, only allows signal monitoring and tracing with your target application.

- 1 In the Simulink window, click **Simulation Parameters**. On the Simulation Parameter dialog box, click the **Real-Time Workshop** tab.

Access levels are set in the **System target file** box. For example, to set the access level to 1, enter

```
xpcrtarget.tlc -axpcWWWAccessLevel=1
```

The effect of not specifying -axpcWWWAccessLevel=# is that the access level of 0 (the highest) is set.

- 2 Click **OK**.

The various fields disappear depending on the access level. For example, if your access level does not allow you access to the parameters, you will not see the button for parameters.

There are various access levels for monitoring, which will allow different levels of hiding. The proposed setup is described below. Each level builds up on the previous one, so only the incremental hiding of each successive level is described

Level 0 - Full access to all pages and functions.

Level 1 - Cannot change the sample and stop times. Cannot change parameters, but can view parameters.

Level 2 - Cannot start and stop execution of the target application.

Level 3 - Cannot view parameters. Cannot add new scopes, but can edit existing scopes.

Level 4 - Cannot edit existing scopes on the Scopes page. Cannot add or remove signals on the Scopes page. Cannot view the Signals page and the Parameters page.

xPC Target Embedded Option

Introduction	5-3
DOSLoader Mode Overview	5-4
StandAlone Mode Overview	5-4
Architecture	5-5
Restrictions	5-6
 Updating the xPC Target Environment	 5-8
 Creating a DOS System Disk	 5-11
 DOS Loader Target Applications	 5-12
Creating a Target Boot Disk for DOS Loader	5-12
Creating a Target Application for DOS Loader	13
 Stand-Alone Target Applications	 5-14
Creating a Target Application for Stand-Alone	5-14
Creating a Target Boot Disk for Stand-Alone	5-15
Using Target Scope Blocks with Stand-Alone	5-15

The xPC Target Embedded Option allows you to boot the target PC from an alternate device other than a floppy disk drive such as a hard disk drive or flash memory. It also allows you to create stand-alone applications on the target PC independent from the host PC.

This chapter includes the following sections:

- **“Introduction”**
- **“Architecture”**
- **“Restrictions”**
- **“Updating the xPC Target Environment”**
- **“Creating a DOS System Disk”**
- **“DOS Loader Target Applications”**
- **“Stand-Alone Target Applications”**

Introduction

The xPC Target Embedded Option allows you to boot the xPC Target kernel from not only a floppy disk drive, but also from other devices, including a flash disk or a hard disk drive. By using xPC Target Embedded Option, you can configure target PCs to automatically start execution of your embedded application for continuous operation each time the system is booted. You use this capability to deploy your own real-time applications on target PC hardware.

The xPC Target Embedded Option extends the xPC Target base product by adding two additional modes of operation:

- **DOSLoader** - This mode of operation is used to start the kernel on the target PC from not only a floppy disk, but optionally start it from a flash disk or a hard disk. The target PC then waits for the host computer to download a real-time application either using the RS232 serial connection or using TCP/IP network communication. Control and setting of starting, stopping, parameters, tracing, and other properties can be achieved from either the host PC or from the target PC.
- **StandAlone** - This mode extends the DOSLoader mode. After starting the kernel on the target PC, StandAlone mode automatically starts execution of your target application for complete stand-alone operation. This eliminates the need for using a host computer and allows you to deploy real-time applications on PC hardware environments.

Whether you are using the xPC Target Embedded Option with the **DOSLoader** mode or the **StandAlone** mode, you initially boot your target PC with DOS from virtually any boot device. Then the kernel is invoked from DOS.

Note The xPC Target Embedded Option requires a boot device with DOS installed. DOS software and license are not included with xPC Target or with the xPC Target Embedded Option.

During setup of either the **DOSLoader** mode or **StandAlone** mode, the xPC Target Setup window allows you to create files for installation on the target boot device. One of these files is an `autoexec.bat` file. When DOS starts, it

invokes the `autoexec. bat` file which in turn starts the xPC Target kernel on the target PC.

If you do not provide an target application and an `autoexec. bat` file to invoke your target application, xPC Target Embedded Option starts the kernel on your target PC and is ready to receive your target application whenever you build and download a new one from the host computer.

In comparison, when using xPC Target without the xPC Target Embedded Option, you can only download real-time applications to the target PC after booting from an xPC Target boot disk. Because of this, when using xPC Target without Embedded Option is not available, you are always required to use a target PC equipped with a floppy disk drive. However, there are several cases where your target system may not have a floppy disk drive or where the drive is removed after setting up the target system. These cases can be overcome by using the DOSLoader mode.

DOSLoader Mode Overview

With the **DOSLoader** mode of operation, you first set up a boot device such as a floppy disk drive, flash disk, or a hard disk drive. This boot device must include DOS and modules from xPC Target Embedded Option. Once the kernel starts running, it awaits commands from the host computer and a target application that is downloaded from the host computer. The primary purpose of the **DOSLoader** mode is to allow you to boot from devices other than the floppy drive.

StandAlone Mode Overview

With the **StandAlone** mode of operation, you create completely stand-alone applications which start execution automatically when the target PC is booted. There is no need for communication with a host computer to download the application after booting. Once the boot device has been set up with DOS, modules from xPC Target Embedded Option, and your target application, you boot the target PC. Upon booting, DOS invokes your `autoexec. bat` file which invokes the kernel. However, in **StandAlone** mode, your target application is combined with kernel in one binary `*.rtb` file. The final result is that your target application starts automatically each time the target PC is booted. By using xPC Target Embedded Option, you can deploy control systems, DSP applications, and other systems on PC hardware for use in production

applications using PC hardware. Typically these production applications are found in systems where production quantities are low to moderate.

xPC Target Embedded Option also gives you the choice of using target scopes on the target PC. When using **StandAlone** mode, target scopes allows you to trace signals using the target PC monitor without any interaction from the host computer. Assuming you do not want to view signals on the target PC, it is not necessary to use target scopes or a monitor on your target PC. In such a case, your system is able to operate as a *black-box* without a monitor, keyboard, or mouse. Stand-alone applications are automatically set to continue running for an infinite time duration or until the target computer is turned off.

Architecture

xPC Target Embedded Option creates additional files that you add to your target PC DOS boot device. With the **DOSLoader** mode, an `autoexec. bat` file is generated. This file enables DOS to automatically execute the file `xpcboot. com` when the target PC is booted. The file `autoexec. bat` includes an argument that invokes a `*. rtb` file containing the xPC Target kernel. Therefore, when the boot device invokes DOS, the `autoexec. bat` file then starts the xPC Target kernel. All of these files are placed on a floppy disk when you click **BootDisk** from the `xpcsetup` GUI. Your real-time application is not copied to the boot device. You create the real-time application later by clicking **Build**.

The **StandAlone** mode operates in a similar fashion with a few important differences. From the `xpcsetup` GUI, after choosing **StandAlone**, you only click **Update** to make your current selections active. When you later click **Build**, an `autoexec. bat` file and the `xpcboot. com` file are placed in a subdirectory that is created within your present working directory. This directory is named: `modename_xpc_emb`. In addition, the build process creates your target application and combines it with the xPC Target kernel. This combined `*. rtb` file is also placed in the same `modename_xpc_emb` subdirectory. You copy these files onto any DOS boot device. Then, upon booting DOS, the `xpcboot.com` file is invoked with the kernel and with your target application. If you choose to use target scopes with your stand-alone application, you can do so provided appropriate xPC Target Scope blocks are added and configured prior to code generation.

A small DOS executable called `xpcboot. com` is the core module of the Embedded Option. This module is used in both the **DOSLoader** mode and the

StandAlone mode. The module `xpcboot.com` is executed from DOS. It loads and executes any xPC Target application. The first argument given to `xpcboot.com` is the name of the image file (*.rtb) to be executed. This image file contains the xPC Target kernel and options, such as whether you are communicating using a serial cable or TCP/IP, and the ethernet address you have assigned to the target PC.

Since the xPC Target loader is just an ordinary xPC Target application, the loader can be executed from `xpcboot.com`.

Before starting the kernel, you must first boot the target PC under DOS. The module `xpcboot.com` is then automatically executed under DOS by `autoexec.bat`. To boot the target PC under DOS, you must first install DOS on the target PC boot device. The xPC Target Embedded Option does not have specific requirements as to the type of device you use to boot DOS. It is possible to boot from a floppy disk drive, hard disk drive, flash disk, or other device where you have installed DOS.

DOS is only needed to execute `xpcboot.com` and read the image file from the file system. After switching to the loaded kernel, and then executing the xPC Target application, DOS is discarded and is unavailable, unless you reboot the target PC without automatically invoking the xPC Target kernel. Once the xPC Target application begins execution, the target application is executed entirely in the protected mode using the 32-bit flat memory model.

Restrictions

The following restrictions apply to the booted DOS environment when you use `xpcboot.com` to execute the target applications:

- The CPU must be executed in real mode
- While loaded in memory, the DOS partition must not overlap the address range of a target application

You can satisfy these restrictions by avoiding the use of additional memory managers like `emmm386` or `qemmm`. Also, you should avoid any utilities that attempt to load in high memory space (for example, `hi mem. sys`). If the target PC DOS environment does not use a `config.sys` file or memory manager entries in the `autoexec.bat` file, there should not be any problems when running `xpcboot.com`.

It is also necessary that your **TargetMouse** setting is consistent with your hardware. Some PC hardware may use an RS232 port for the mouse, while others use a PS2 mouse. If a mouse is not required in your application, you may choose to select **None** as your setting for the **TargetMouse**.

Updating the xPC Target Environment

After the xPC Target Embedded Option software has been correctly installed, the xPC Target environment, visible through `xpcsetup` or `getxpcenv`, contains new property choices for **DOSLoader** or **StandAlone**, in addition to the default **BootDisk** that is normally used with xPC Target.

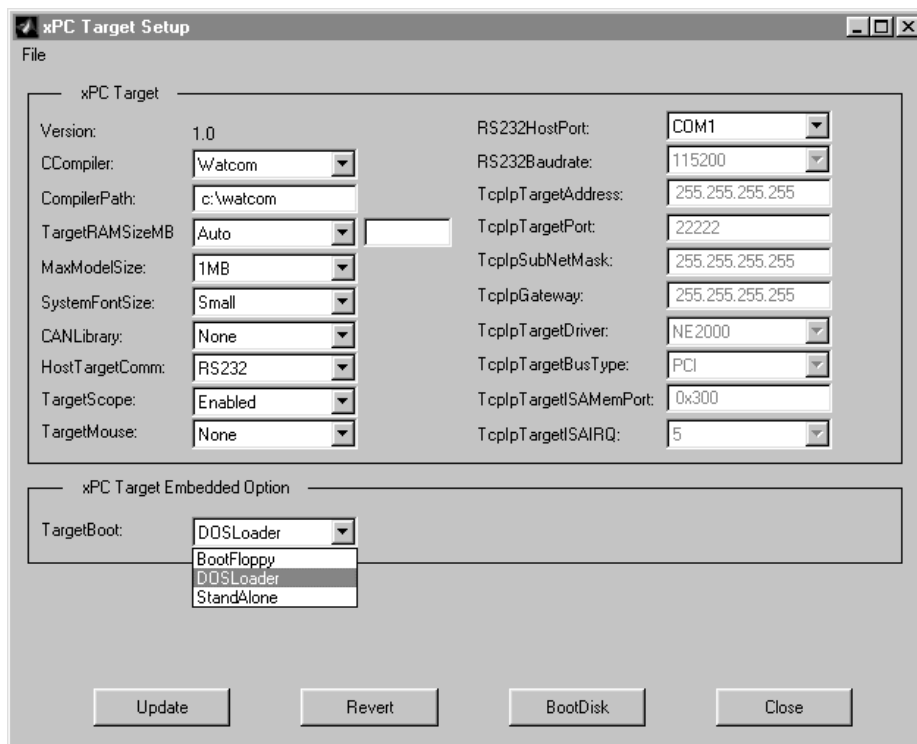
It is assumed that the xPC Target environment is already set up and working properly with the xPC Target Embedded Option disabled. If you have not already done so, we recommend you confirm this now.

You can use the function `getxpcenv` to see the current selection for **TargetBoot**, or you can view this through the xPC Target Setup window. Start MATLAB and execute the function

```
xpcsetup
```

Within the frame of the xPC Target Embedded Option, you see the property **TargetBoot**, as well as the currently selected value. The choices are:

- **BootFloppy** - Standard mode of operation when xPC Target Embedded Option is not installed.
- **DOSLoader** - For invoking the kernel on the target PC from DOS.
- **StandAlone** - For invoking the kernel on the target PC from DOS and automatically starting the target application without connecting to a host computer. With this mode, the kernel and the target application are combined as a single module that is placed on the boot device.



The default setting for the property **TargetBoot** is **BootFloppy**. When using **BootFloppy**, xPC Target must first create a target boot disk, which is then used to boot the target PC.

The property **TargetBoot** can be set to two other values, namely **DOSLoader** or **StandAlone**. If the xPC Target loader is booted from any boot device with DOS installed, the value **DOSLoader** must be set as shown above. If you want to use a stand-alone application that automatically starts execution of your target application immediately after booting, specify **StandAlone**.

After changing the property value, you need to update the xPC Target environment by clicking the **Update** button in the xPC Target Setup window. If your choice is **DOSLoader**, a new target boot disk must be created by clicking **BootDisk**. Note that this overwrites the data on the inserted target boot disk as new software modules are placed on the target boot disk. If your choice is **StandAlone**, you click the **Update** button. Upon building your next real-time application, all necessary xPC Target files are saved to a subdirectory below your present working directory. This subdirectory is named with your model name with the string “_xpc_emb” appended. For example, xpcosc_xpc_emb.

For more detailed information about how to use the xPC Target Setup, see Chapter 6, “Environment Reference.”

Creating a DOS System Disk

When using DOSLoader mode, or StandAlone mode, you must first boot your target PC with DOS. These modes may be used from any boot device including flash disk, floppy disk drive, or a hard disk drive.

In order to boot DOS with a target boot disk, a minimal DOS system is required on the boot disk. With Windows 95, Windows 98, or DOS, you can create a DOS boot disk using the command

```
sys a:
```

Note xPC Target Embedded Option does not include a DOS license. You must obtain a valid DOS license for your target PC.

It is helpful to also copy additional DOS-utilities to the boot disk including

- A DOS-editor to edit files
- The format program to format a hard disk or FlashRAM
- The fdisk program to create partitions
- The sys program to transfer a DOS system onto another drive, such as the hard disk drive

A config.sys file is not necessary. The autoexec.bat file should be created to automatically boot the loader or a standalone xPC Target application. This is described in the following sections.

DOS Loader Target Applications

This section includes the following topics:

- “Creating a Target Boot Disk for DOS Loader”
- “Creating a Target Application for DOS Loader”

Creating a Target Boot Disk for DOS Loader

As the first step, we assume you have created a DOS system disk and updated the xPC Target environment by setting the property **TargetBoot** to **DOSLoader**. From the xPC Target Setup window, click the **BootDisk** button and xPC Target copies the necessary files to the DOS disk. The files that are added to the DOS boot disk include:

- checksum.dat
- autoexec.bat
- xpcboot.com
- *.rtb (where * is defined in the table below)

With the **DOSLoader** mode, the correct *.rtb file is added to the DOS disk according to the options specified in the following table.

Table 5-1: DOSLoader Mode *.rtb File Naming Convention

xPC Target environment	HostTargetComm: RS232	HostTargetComm: TCP/IP
TargetScope: Disabled	xpcston.rtb	xpctton.rtb
TargetScope: Enabled	xpcsgon.rtb	xpctgon.rtb

Note The numeric value of *n* corresponds to the maximum model size. This value is either 1, 4, or 16 megabytes. The default value for *n* is 1, or a 1-megabyte maximum model size.

The file `autoexec.bat` is copied to the DOS disk. This file should contain at least the following line:

```
xpcboot xxx.rtb
```

where `xxx.rtb` is the file described in table 11-1. We recommend that you view this `autoexec.bat` file to confirm this.

Now the target boot disk can be removed from the host and put into the target PC disk drive. By rebooting the target PC, DOS is booted from the target boot disk and the `autoexec.bat` file with the result in the xPC Target loader being automatically executed. From this point onwards, the CPU runs in protected mode and DOS is discarded.

You can repeat this procedure as necessary. There are no restrictions on the number of xPC Target boot floppies that you can create. However, xPC Target and the xPC Target Embedded Option do not include DOS licenses. It is assumed that you will purchase valid DOS licenses for your target PCs from the supplier of your choice.

If the `xpcboot` command is not placed in the `autoexec.bat`, `xpcboot.com` is not executed when the target PC is booted. Instead, the target will be finished once it has booted DOS. You can then use the DOS environment to create a DOS partition on a hard disk, format it, and transfer `xpcboot.com` and `xxx.rtb` onto it. The `autoexec.bat` file can then be placed on the hard disk and edited so that it automatically boots the xPC Target loader the next time the target PC is booted. After this step the floppy disk drive can be removed from the system. The same procedure works with flash disks and other boot devices.

Creating a Target Application for DOS Loader

After having booted the target PC as described in the proceeding section, the target PC is ready to receive xPC Target applications from the host computer. Only now, these applications are received by the **DOSLoader** component of xPC Target. In every aspect, the **DOSLoader** mode will allow your target PC to operate just as it normally would when running the xPC Target after booting from a standard xPC Target boot disk. When you click **Build** for your model, the target application is downloaded to the target PC using the communication protocol as you specified earlier in the xPC Target Setup window.

Stand-Alone Target Applications

This section includes the following topics:

- “**Creating a Target Application for Stand-Alone**”
- “**Creating a Target Boot Disk for Stand-Alone**”
- “**Using Target Scope Blocks with Stand-Alone**”

Creating a Target Application for Stand-Alone

After selecting **StandAlone** as your **TargetBoot** entry, the xPC Target environment is ready to create completely stand-alone applications using the Real-Time Workshop **Build** button.

Once the build process has finished, a message is displayed confirming that a stand-alone application has been created. With the **StandAlone** mode, the download procedure is not automated. The files necessary for creating stand-alone operation are placed in a subdirectory below your working directory. You copy these files to your DOS boot device.

After the build process is complete, files in your subdirectory include:

- `model . rtb`. This image contains the xPC Target kernel and your target application.
- `autoexec. bat`. This file is automatically invoked by DOS. It then runs `xpcboot. com` and the `*. rtb` file.
- `xpcboot. com`. This file is a static file that is part of xPC Target Embedded Option.

Note We suggest setting the property **HostTargetComm** to **RS232** if property **TargetBoot** is set to **StandAlone**. This will use less memory than the TCP/IP setting. With either setting, **StandAlone** mode does not have any interaction with the host PC.

Creating a Target Boot Disk for Stand-Alone

After making a bootable DOS boot disk, the file `autoexec.bat` file must contain at least the following line

```
xpcboot model.rtb
```

where *model* is the name of your Simulink model.

These files should be copied to your DOS boot disk and inserted into the target drive. By rebooting the target PC, DOS is booted from the boot disk. The `autoexec.bat` file invokes the command string shown above which starts the kernel and the real-time application. Because of the stand-alone nature of the executed `rtb` file, the simulation of the xPC Target application starts immediately. Interaction between the host PC and target PC is no longer possible.

Is also possible to transfer the DOS system and stand-alone xPC Target applications to a hard disk or a flash RAM board. This offers great flexibility in creating self-starting stand-alone applications.

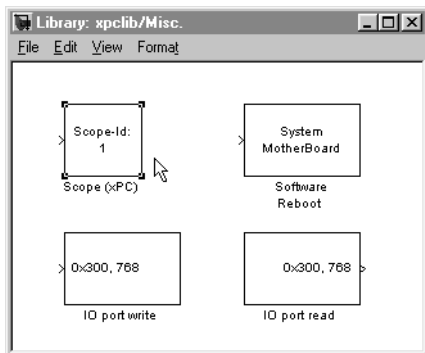
Using Target Scope Blocks with Stand-Alone

When using xPC Target Embedded Option with **StandAlone** mode, you can also use target scopes for tracing signals and displaying them on the target screen.

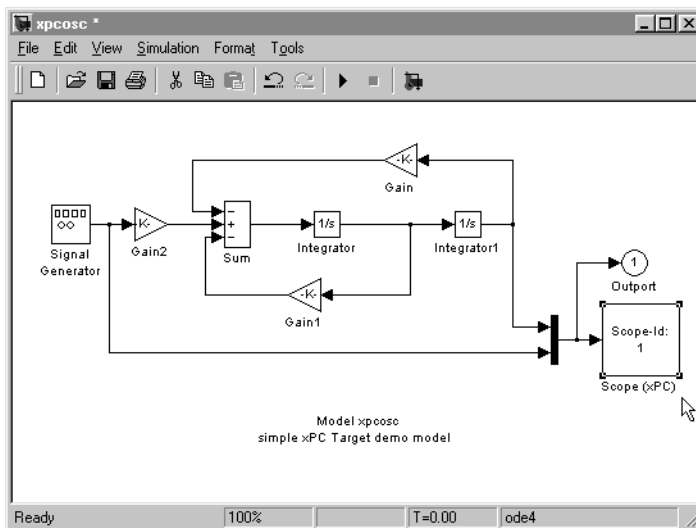
Because host-to-target communication is not supported with the **StandAlone** mode, scope objects of type `target` must be defined within the Simulink model before the xPC Target application is built.

xPC Target Embedded Option

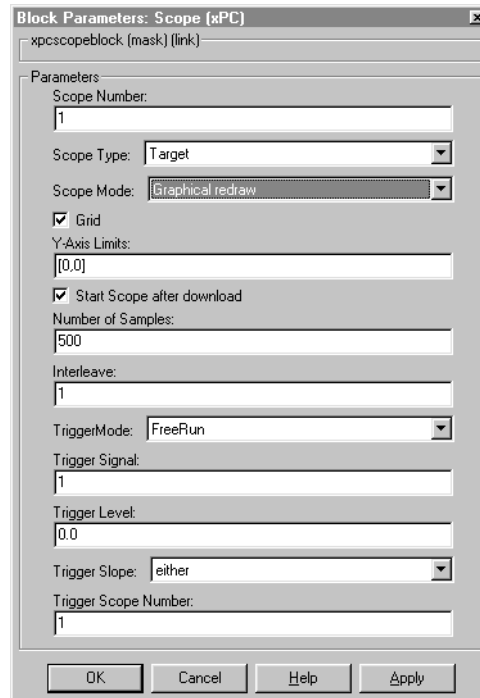
xPC Target (basic package) offers a block for such purposes.



Copy the **Scope (xPC)** block into your block diagram and connect the signals you would like to view to this block. Multiple signals can be used provided a **Mux** block is used to bundle them.



It is necessary to edit the **Scope (xPC)** dialog box and confirm that the check box entry for **Start Scope after download** is checked as shown in the following dialog box.



This setting is required to enable target scopes to begin operating as soon as the application starts running. The reason this setting is required is that the host PC is not available in StandAlone mode to issue a command that would start scopes. With these settings, click **Build** and copy the files from your *model name_xpc_emb* subdirectory to your boot disk. Then boot your target PC. When the target application starts to run, the target scopes will start automatically. A monitor is needed on your target PC to view the results.

Note When using target scopes with **StandAlone** mode, you must specify the Scope Type as Target prior to generating code.

Environment Reference

Environment	6-3
Environment Properties	6-3
Environment Functions	6-11
 Using Environment Properties and Functions	 6-12
Getting a List of Environment Properties	6-12
Saving and Loading the Environment	6-13
Changing Environment Properties with Graphical Interface	6-14
Changing Environment Properties with Command Line Interface	6-16
Creating a Target Boot Disk with Graphical Interface . . .	6-17
Creating a Target Boot Disk with Command Line Interface .	6-19
 System Functions	 6-20
GUI Functions	6-20
Test Functions	6-21
xPC Target Demos	6-21
Environment and System Function Reference	6-23

The xPC Target environment defines connections and communication between the host and target computers. It also, defines the build process for a real-time application.

This chapter includes the following sections:

- **“Environment”** - List of environment properties and functions with a brief description
- **“Using Environment Properties and Functions”** - Common tasks within the xPC Target software environment
- **“System Functions”** - List of functions for testing and opening graphical interfaces

Environment

The xPC Target environment defines the software and hardware environment of the host PC as well as the target PC. An understanding of the environment properties will help you to correctly configure the xPC Target environment.

This section includes the following topics:

- “**Environment Properties**” - List of properties with a brief description
- “**Environment Functions**” - List of functions with a brief description

Environment Properties

The environment properties define communication between the host PC and target PC, the type of C compiler and its location, and the type of target boot floppy created during the setup process. You can view and change these properties using the environment functions or Setup window.

Table 6-1: List of Environment Properties

Environment property	Description
Versi on	xPC Target version number. Read-only.
Path	xPC Target root directory. Read-only.
CCompi l er	Values are ' Watcom' or ' Vi sual C' . From the Setup window CCompiler list, choose either Watcom or VisualC .
Compi l erPath	Value is a valid compiler root directory. Enter the path where you installed a Watcom C/C++ or Microsoft Visual C/C++ compiler. If the path is invalid or the directory does not contain the compiler, then when you use the function updatexpcenv or build a target application, an error message appears.

Table 6-1: List of Environment Properties

Environment property	Description
TargetRAMSizeMB	<p>Values are 'Auto' or 'MB of target RAM'.</p> <p>From the Setup window TargetRAMSizeMB list, choose either Auto or Manual. If you select Manual, enter the amount of RAM, in megabytes, installed on the target PC. This property is set by default to Auto.</p> <p>TargetRAMSizeMB defines the total amount of installed RAM in the target PC. This RAM is used for the, kernel, target application, data logging, and other functions that use the heap.</p> <p>If TargetRAMSizeMB is set to Auto, the target application automatically determines the amount of memory up to 64 MB. If the target PC does not contain more than 64 MB of RAM, or you do not want to use more than 64 MB, select Auto. If the target PC has more than 64 MB of RAM, and you want to use more than 64 MB, select Manual, and enter the amount of RAM installed in the target PC.</p>
MaxModelSize	<p>Values are '1MB', '4MB', or '16MB'.</p> <p>From the Setup window MaxModelSize list, choose either 1 MB, 4 MB, or 16 MB.</p> <p>Choosing the maximum model size reserves the specified amount of memory on the target PC for the target application. The remaining memory is used by the kernel and by the heap for data logging.</p> <p>Selecting too high a value leaves less memory for data logging. Selecting too low a value does not reserve enough memory for the target application and creates an error.</p>

Table 6-1: List of Environment Properties

Environment property	Description
SystemFontSize	<p>Values are ' Small ' or ' Large ' .</p> <p>From the Setup window SystemFontSize list, choose either Small, or Large.</p> <p>The xPC Target GUIs use this information to change the font size.</p>
CANLibrary	<p>Values are ' None ' , ' 200 ISA ' , ' 527 ISA ' , ' 1000 PCI ' , ' 1000 MB PCI ' , or ' PC104 ' .</p> <p>From the Setup window CANLibrary list, choose None, 200 ISA, 527 ISA, 1000 PCI, 1000 MB PCI, or PC104.</p>
HostTargetComm	<p>Values are ' RS232 ' or ' TcpIp ' .</p> <p>From the Setup window HostTargetComm list, choose either RS232 or TCP/IP.</p> <p>If you select RS232, you also need to set the property RS232HostPort. If you select TCP/IP, then you also need to set all properties that start with TcpIp.</p>
RS232HostPort	<p>Values are ' COM1 ' or ' COM2 ' .</p> <p>From the Setup window RS232HostPort list, choose either COM1 or COM2 for the connection on the host computer. xPC Target automatically determines the COM port on the target PC.</p> <p>Before you can choose an RS232 port, you need to set the HostTargetComm property to RS232.</p>
RS232Baudrate	<p>Values are ' 115200 ' , ' 57600 ' , ' 38400 ' , ' 19200 ' , ' 9600 ' , ' 4800 ' , ' 2400 ' , or ' 1200 ' .</p> <p>From the RS232Baudrate list, choose 115200, 57600, 38400, 19200, 9600, 4800, 2400, or 1200.</p>

Table 6-1: List of Environment Properties

Environment property	Description
TcpIpTargetAddress	<p>Value is ' xxx. xxx. xxx. xxx' .</p> <p>In the Setup window TcpIpTargetAddress box, enter a valid IP address for your target PC. Ask you system administrator for this value.</p> <p>For example, 192. 168. 0. 1</p>
TcpIpTargetPort	<p>Value is ' xxxxx' .</p> <p>In the Setup window TcpIpTargetPort box, enter a value greater than 20000.</p> <p>This property is set by default to 22222 and should not cause any problems. The number is higher than the reserved area (telnet, ftp, ...) and it is only of use on the target PC.</p>
TcpIpSubNetMask	<p>Value is ' xxx. xxx. xxx. xxx' .</p> <p>In the Setup window TcpIpSubNetMask text box, enter the subnet mask of your LAN. Ask you system administrator for this value.</p> <p>For example, 255. 255. 0. 0.</p>

Table 6-1: List of Environment Properties

Environment property	Description
TcpIpGateway	<p>Value is ' xxx. xxx. xxx. xxx' .</p> <p>In the Setup window TcpIpGateway box, enter the IP address for your gateway. This property is set by default to 255. 255. 255. 255 whi ch means that a gateway is not used to connect to the target PC.</p> <p>If you communicate with your target PC from within a LAN that uses gateways, and your host and target computers are connected through a gateway, then you need to enter a value for this property. If your LAN does not use gateways, you do not need to change this property. Ask your system administrator.</p>
TcpIpTargetDi rver	<p>Values are ' NE2000' or ' SMC91C9X' .</p> <p>From the Setup window TcpIpTargetDriver list, choose either NE2000 or SMC91C9X. The Ethernet card provided with xPC Target uses the NE2000 driver.</p>
TcpIpTargetBusType	<p>Values are ' PCI' or ' ISA' .</p> <p>From the Setup window TcpIpTargetBusType list, choose either PCI or ISA. This property is set by default to PCI, and determines the bus type of your target PC. You do not need to define a bus type for your host PC, which can be the same or different from the bus type in your target PC.</p> <p>If TcpIpTargetBusType is set to PCI, then the properties TcpIpISAMemPort and TcpIpISAIRQ have no effect on TCP/IP communication.</p> <p>If you are using an ISA bus card, set TcpIpTargetBusType to ISA and enter values for TcpIpISAMemPort and TcpIpISAIRQ.</p>

Table 6-1: List of Environment Properties

Environment property	Description
TcpIpTargetISAMemPort	<p>Value is '0xnnnn' .</p> <p>If you are using an ISA-bus Ethernet card, you must enter values for the properties TcpIpISAMemPort and TcpIpISAIRQ. The values of these properties must correspond to the jumper settings or ROM settings on your ISA-bus Ethernet card.</p> <p>On your ISA-bus card, assign an IRQ and I/O-port base address by moving the jumpers on the card.</p> <p>We recommend setting the I/O-port base address to around 0x300. If one of these hardware settings leads to a conflict in your target PC, choose another I/O-port base address and make the corresponding changes to your jumper settings.</p>
TcpIpTargetISAIRQ	<p>Value is ' n' where <i>n</i> is between 4 and 15.</p> <p>If you are using an ISA bus Ethernet card, you must enter values for the properties TcpIpISAMemPort and TcpIpISAIRQ. The values of these properties must correspond to the jumper settings or ROM settings on the ISA-bus Ethernet card.</p> <p>On your ISA-bus card, assign an IRQ and I/O-port base address by moving the jumpers on the card.</p> <p>We recommend setting the IRQ to 5, 10, or 11. If one of these hardware settings leads to a conflict in your target PC, choose another IRQ and make the corresponding changes to your jumper settings.</p>

Table 6-1: List of Environment Properties

Environment property	Description
EmbeddedOption	<p>Values are 'Disabled' or 'Enabled'. This property is read-only.</p> <p>Note The xPC Target Embedded Option is enabled only if you purchase an additional license.</p>
TargetScope	<p>Values are 'Disabled' or 'Enabled'.</p> <p>From the Setup window TargetScope list, choose either Enabled or Disabled.</p> <p>The property TargetScope is set by default to Enabled. If you set TargetScope to Disabled, then the target PC displays information as text.</p> <p>To use all of the features of target scope, you also need to install a keyboard and mouse on the target PC.</p>

Table 6-1: List of Environment Properties

Environment property	Description
TargetMouse	<p>Values are 'None', 'PS2', 'RS232 COM1', 'RS232 COM2'.</p> <p>From the Setup window TargetMouse list, choose None, PS2, RS232 COM1, or RS232 COM2.</p> <p>Before you can select a target mouse, you need to set the Target Scope property to Enabled.</p> <p>TargetMouse allows you to disable or enable mouse support on the target PC:</p> <ul style="list-style-type: none">• If you do not connect a mouse to the target PC, you need to set this property to None, otherwise, the target application may not behave properly.• If the target PC supports PS/2 devices (keyboard and mouse) and you connect a PS/2 mouse, set this property to PS2.• If you connect a serial RS232 mouse to the target PC, choose either RS232 COM1 or RS232 COM2 depending on which serial port you attached the mouse.
TargetBoot	<p>Values are 'BootFloppy', 'DOSLoader', or 'StandAlone'.</p> <p>From the Setup window TargetBoot list, choose either BootFloppy, DOSLoader, or StandAlone.</p> <p>If your license file does not include the license for the xPC Target Embedded Option, the Target Boot box is disabled with BootFloppy as your only selection. With the xPC Target Embedded Option licensed and installed, you have the additional choices of DOSLoader and StandAlone.</p>

Environment Functions

The environment functions allow you to change the environment properties. The functions are listed in the following table.

Table 6-2: List of Environment functions

Environment functions	Description
getxpcenv	List environment properties in the MATLAB window or assign the list as a cell array to a MATLAB variable.
setxpcenv	First of two steps to change environment properties. See also updatexpcenv.
updatexpcenv	Changes the current environment properties to equal the new properties entered using the function setxpcenv.
xpcbootdisk	Creates a boot floppy disk containing the kernel according to the current environment properties.

Using Environment Properties and Functions

You use the xPC Target Setup window to enter properties that are independent of your model.

This section includes the following topics:

Changing Environment Properties

- **“Getting a List of Environment Properties”**
- **“System Functions”**
- **“Changing Environment Properties with Graphical Interface”**
- **“Changing Environment Properties with Command Line Interface”**

Target Boot Disk

- **“Creating a Target Boot Disk with Graphical Interface”**
- **“Creating a Target Boot Disk with Command Line Interface”**

To enter properties specific to your model and its build procedure, see “Entering the Simulation Parameters” on page 3-8. These properties are saved with your Simulink model.

Getting a List of Environment Properties

To use the xPC Target functions to change environment properties, you need to know the names and allowed values of the properties. Use the following procedure to get a list of the property names, their allowed values, and their current values:

- 1 In the MATLAB window, type

```
setxpcenv
```

MATLAB displays a list of xPC Target environment properties and the allowed values. For a list of the properties, see “Environment Properties” on page 6-3

- 2 Type

```
getxpcenv
```

MATLAB displays a list of xPC Target environment properties and the current values.

- 3 Alternately, in the MATLAB window, type
`xpcsetup`

MATLAB opens the xPC Target Setup window with the current values.

Saving and Loading the Environment

This feature makes it easy and fast to switch between different xPC Target environments:

- 1 In the xPC Target Setup window, and from the **File** menu, click **Save Settings**.

The Save xPC Target Environment dialog box opens.

- 2 Enter the name of an environment file (*.mat). Select a directory, and then click **Save**.

xPC Target saves the current environment properties.

After you have saved an xPC Target environment, you can load those property values back into xPC Target.

- 3 From the **File** menu, click **Load Settings**.

The Load xPC Target Environment dialog box opens.

- 4 Select a directory with a previously saved environment file (*.mat). Select the file, and then click **Open**.

- 5 In the xPC Target Setup window, click the **Close** button.

If you changed the environment properties, but do not click the Update button, xPC Target displays a warning.

Even if you decide to continue with the exit process, you will not lose the values you changed. However, the current environment does not reflect the changes you made in the xPC Target Setup window. If you reopen the xPC Target Setup window, the changes you made reappear, and the **Update** button is enabled.

Changing Environment Properties with Graphical Interface

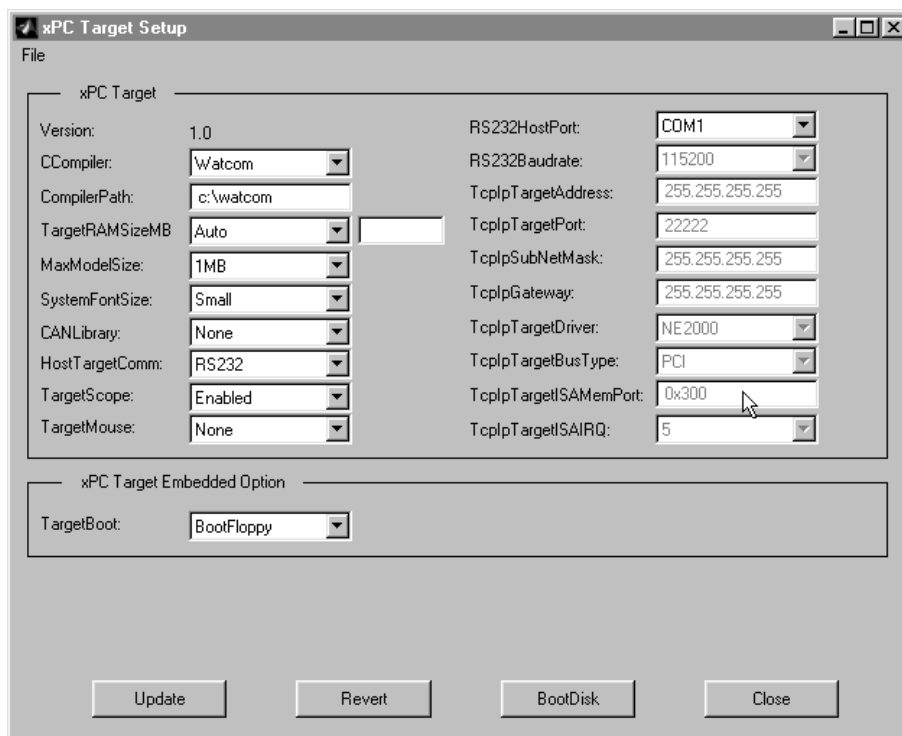
xPC Target lets you define and change environment properties. These properties include, the path to the C/C++ compiler, the host PC COM-port, the logging buffer size, and many others. Collectively these properties are known as the xPC Target environment.

To change an environment property using the xPC Target GUI, use the following procedure:

- 1 In the MATLAB command window, type

```
xpcsetup
```

MATLAB opens the xPC Target Setup window.



The xPC Target Setup window has two sections:

- **xPC Target**
- **xPC Target Embedded Option**

If your license does not include the xPC Target Embedded Option, the **TargetBoot** box is grayed-out with **BootFloppy** as your only selection. With the xPC Target Embedded Option, you have the additional choices of **DOSLoader** and **StandAlone**.

- 2 Change properties in the environment by entering new property values in the text boxes or choosing items from the lists.

After you make changes to the environment properties, you need to update the xPC Target environment. Updating makes your changes in the xPC Target Setup window equal to the current property values.

- 3 Click the **Update** button.

xPC Target updates the xPC Target environment and disables (grays-out) the **Update** button. As long as the **Update** button is enabled, the xPC Target environment needs to be updated.

Changing Environment Properties with Command Line Interface

xPC Target lets you define and change different properties. These properties include, the path to the C/C++ compiler, the host COM-port, the logging buffer size, and many others. Collectively these properties are known as the xPC Target environment.

You can use the command line functions to write an M-file script that accesses the environment settings according to your own needs. For example, you could write an M-file that switches between two targets.

The following procedure shows how to change the COM port property for your host PC from COM1 to COM2:

- 1 In the MATLAB window, type

```
setxpcenv(' RS232HostPort' , ' COM2' )
```

The up-to-date column shows the values that you have changed, but have not updated.

HostTargetComm	: RS232	up to date
RS232HostPort	: COM2	COM2
RS232Baudrate	: 115200	up to date

Making changes using the function `setxpcenv` does not change the current values until you enter the update command.

- 2 In the MATLAB window, type

```
updatexpcenv
```

The environment properties you changed with the function `setxpcenv` become the current values.

HostTargetComm	: RS232	up to date
RS232HostPort	: COM2	up to date
RS232Baudrate	: 115200	up to date

Creating a Target Boot Disk with Graphical Interface

You use the target boot disk to load and run the xPC Target kernel.

After you make changes to the xPC Target environment properties, you need to create or update a target boot disk. To create a target boot disk for the current xPC Target environment, use the following procedure:

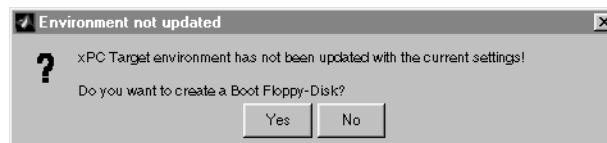
- 1 In the MATLAB window, type

```
xpcsetup
```

The xPC Target Setup window opens.

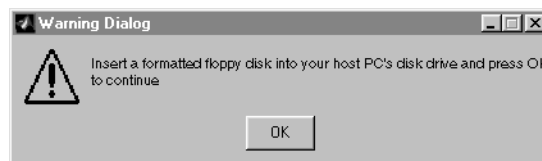
- 2 Click the **BootDisk** button.

If you didn't update the current settings, the following message box appears.



Click No. Click the Update button, and then click the BootDisk button again.

After you update the current properties, and click the **BootDisk** button, the following message box appears.



- 3 Insert a formatted floppy disk into the host PC disk drive, and then click **OK**. All data on the disk will be erased.

The write procedure starts and while creating the boot disk, the MATLAB command window displays the following status information. On Windows

NT systems, the status information is displayed only at the end of the write process.

xPC Target DiskWrite Utility Version 1.1, (c) 1998-2000 The MathWorks, Inc. Read File:

```
C:\MATLAB\TOOLBOX\RTW\TARGETS\XPC\XPC\BIN\
..\..\target\kernel\xpccsgb1.rtd
```

[illegible]

The process of creating a boot disk takes about 1 to 2 minutes.

- 4 Close the xPC Target Setup window.
- 5 Remove the target boot disk from the host PC disk drive. You may now use this disk to boot your target PC.

To create a boot disk using the command line interface, see “Creating a Target Boot Disk with Command Line Interface” on page 6-19.

Creating a Target Boot Disk with Command Line Interface

You use the target boot disk to load and run the xPC Target kernel.

After you make changes to the xPC Target environment properties, you need to create or update a boot disk. To create a target boot disk for the current xPC Target environment, use the following procedure:

- 1 In the MATLAB window, type

```
xpcbootdisk
```

xPC Target displays the following message.

```
Insert a formatted floppy disk into your host PC's
disk drive and press any key to continue.
```

- 2 Insert a formatted floppy disk into the host PC disk drive, and then press any key.

The write procedure starts and while creating the boot disk, the MATLAB command window displays the following status information. On Windows NT systems, the status information is displayed only at the end of the write process.

```
xPC Target Disk Write Utility Version 1.1,
(c) 1998-1999 The MathWorks Inc.
Read File: C:\MATLAB\TOOLBOX\RTW\TARGETS\XPC\XPC\BIN\..\
\target\kernel\xpccsgb1.rtd
```

```
Write Track 0- 9: .....
Write Track 10-19: .... uuuuuuuuuuuuuuuuuu
Write Track 20-29: uuuuuuuuuuuuuuuuuuuuuu
Write Track 30-39: uuuuuuuuuuuuuuuuuuuuuu
Write Track 40-49: uuuuuuuuuuuuuuuuuuuuuu
Write Track 50-59: uuuuuuuuuuuuuuuuuuuuuu
Write Track 60-69: uuuuuuuuuuuuuuuuuuuuuu
Write Track 70-79: uuuuuuuuuuuuuuuuuuuuuu
```

To create a boot disk using the graphical interface, see “Creating a Target Boot Disk with Graphical Interface” on page 6-17.

System Functions

The system functions allow you to open xPC Target GUIs and run tests from the MATLAB window.

This section includes the following topics:

- “GUI Functions”
- “Test Functions”
- “xPC Target Demos”

GUI Functions

The GUI functions are listed in the following table.

Table 6-3: List of GUI Functions

System functions	Description
xpcscope	Opens the scope manager window on the host PC for scopes with type host.
xpcsetup	Opens the Setup window.
xpctargetspy	Open the Target Spy window on the host PC. Use this GUI to upload the target PC screen to the host PC.
xpctest	Test the xPC Target installation.
xpcscope	Opens the scope manager window on the host PC for scopes with type host.
xpcsetup	Opens the Setup window.
xpctargetspy	Open the Target Spy window on the host PC. Use this GUI to upload the target PC screen to the host PC.
xpctest	Test the xPC Target installation.

Test Functions

The test functions are listed in the following table.

Table 6-4: List of Test Function

System functions	Description
getxpcpci	Determine which PCI boards are installed in the target PC.
xpctargetpi ng	Test the communication between the host PC and the target PC
xpctargetspy	Open the Target Spy window on the host PC. Use this GUI to upload the target PC screen to the host PC.
xpctest	Test the xPC Target installation.

xPC Target Demos

The xPC Target demos are used to demonstrate the features of xPC Target. But they are also M-file scripts that you can view to understand how to write your own scripts for creating and testing target applications.

The following table lists the demo scripts that we provide with xPC Target.

Demo	Filename
Parameter Sweep	parsweepdemo
Signal tracing using free-run mode	scfreerundemo
Signal tracing using software triggering	scsoftwaredemo
Signal tracing using signal triggering	scsignal demo
Signal tracing using scope triggering	scscopdemo
Signal tracing using the target scope.	tgscopdemo

To locate or edit a demo script

- 1 In the MATLAB window, type

```
scfreerundemo
```

MATLAB displays the location of the M-file.

```
D: \MATLAB\tool box\rtw\targets\xpc\xpcdemos\scfreerundemo.m
```

- 2 Type

```
edit scfreerundemo
```

MATLAB opens the M-file in a MATLAB editing widow.

Environment and System Function Reference

This section includes an alphabetical listing of the environment and system functions.

For a list of the functions with a brief description, see:

- “Environment Functions” on page 6-11
- “GUI Functions” on page 6-20
- “Test Functions” on page 6-21

getxpcenv

Purpose	List environment properties assign to a MATLAB variable
Syntax	MATLAB Command Line getxpcenv
Description	Function for environment properties. This function displays, in the MATLAB window, the property names, the current property values, and the new property values set for the xPC Target environment.
Examples	<p>Return the xPC Target environment in the structure shown below. The output in the MATLAB window is suppressed. The structure contains three fields for property names, current property values, and new property values.</p> <pre>env =getxpcenv</pre> <pre>env = propname: {1x25 cell} actpropval: {1x25 cell} newpropval: {1x25 cell}</pre> <p>Display a list of the environment property names, current values, and new values.</p> <pre>env =getxpcenv</pre>
See Also	The xPC Target functions setxpcenv, updatexpcenv, xpcbootdisk, and xpcsetup.

Purpose	Determine which PCI boards are installed in the target PC		
Syntax	MATLAB Command Line getxpcpci (' type_of_boards')		
Arguments	<table><tr><td>type_of_boards</td><td>Values are no arguments, 'all' and 'supported'.</td></tr></table>	type_of_boards	Values are no arguments, 'all' and 'supported'.
type_of_boards	Values are no arguments, 'all' and 'supported'.		
Description	<p>The information is displayed in the MATLAB window. Only devices supported by driver blocks in the xPC Target Block Library are displayed. The information includes the PCI bus number, slot number, assigned IRQ number, manufacturer name, board name, device type, manufacturer PCI Id, and the board PCI Id itself.</p> <p>For a successful query:</p> <ul style="list-style-type: none">• The host-target communication link must be working. (The function xpctargetping must return success before using the function getxpcpci .• Either a target application is loaded or the loader is active. The latter is used to query for resources assigned to a specific PCI device, which have to be provided to a driver block dialog box prior to the model build process.		
Examples	<p>Return the result of the query in the struct pci devs instead of displaying it. The struct pci devs is an array with one element for each detected PCI device. Each element combines the information by a set of fieldnames. The struct contains more information compared to the displayed list, such as the assigned base addresses, the base and sub class.</p> <pre>pci devs = getxpcpci</pre> <p>Display the supported and installed PCI devices.</p> <pre>getxpcpci (' all')</pre> <p>Display the installed PCI devices, not only the devices supported by the xPC Target Block Library. This will include graphics controller, network cards, SCSI cards and even devices which are part of the motherboard chipset (for example PCI-to-PCI bridges).</p> <pre>getxpcpci (' all')</pre>		

getxpcpci

Display a list of the currently supported PCI devices in the xPC Target block library. The result is stored in a struct instead of displaying it.

```
getxpcpci ( ' supported' )
```

Purpose	Change xPC Target environment properties.	
Syntax	MATLAB Command Line setxpcenv(' property_name' , ' property_val ue') setxpcenv(' prop_name1' , ' prop_val 1' , ' prop_name2' , prop_val 2') setxpcenv	
Arguments	property_name	Not case sensitive. Property names can be shortened as long as they can be differentiated from the other property names.
	property_val ue	Character string. Type setxpcenv without arguments to get a listing of allowed values. Property values are not case sensitive.
Description	<p>Function for environment properties. Enter new environment properties. If the new value is different from the current value, the property is marked as having a new values. Use the function updatexpcenv to change the current properties to the new properties.</p> <p>The function setxpcenv works similarly to the function set of the MATLAB Handle Graphics system. The function setxpcenv must be called with an even number of arguments. The first argument of a pair is the property name, and the second argument is the new property value for this property.</p> <p>Using the function setxpcenv without arguments returns a list of allowed property values in the MATLAB window.</p>	
Examples	<p>List the current environment properties. For a description of properties and allowed values, see “Environment Properties” on page 6-3.</p> <pre>setxpcenv</pre> <p>Change the host PC, serial communication port, to COM2.</p> <pre>setxpcenv(' HostCommPort' , ' COM2')</pre>	
See Also	The xPC Target functions getxpcenv, updatexpcenv, xpcbootdi sk, and xpcsetup. The procedures “Changing Environment Properties with Graphical	

Interface” on page 6-14 and “Changing Environment Properties with Command Line Interface” on page 6-16.

Purpose	Change current environment properties to equal new properties
Syntax	MATLAB Command Line updatexpcenv
Description	Function for environment properties. This procedure includes creating communication M-files as well as patching the xPC Target kernel and system DLL's. Calling the function updatexpcenv is necessary after new properties are entered with the function setxpcenv, but before creating a target boot floppy with the function xpcbootdisk.
See Also	The xPC Target functions setxpcenv, getxpcenv, updatexpcenv, xpcbootdisk, and xpcsetup. The procedures "Changing Environment Properties with Graphical Interface" on page 6-14 and "Changing Environment Properties with Command Line Interface" on page 6-16.

xpcbootdisk

Purpose	Create xPC Target boot disk, and confirm the current environment properties
Syntax	MATLAB Command Line xpcbootdisk
Description	<p>Function for environment properties. This function creates a xPC target boot floppy for the current xPC Target environment which has been updated with the function <code>updatepcenv</code>. Creating an xPC Target boot floppy consists of writing the correct bootable kernel image onto the disk. You are asked to insert an empty, formatted floppy disk into the floppy drive.</p> <p>All existing files are erased by the function <code>xpcbootdisk</code>. If the inserted floppy disk already is an xPC Target boot disk for the current environment settings, this function exits without writing a new boot image to the floppy disk. At the end, a summary of the creation process is displayed.</p> <p>If you update the environment, you need to update the target boot floppy for the new xPC environment with the function <code>xpcbootdisk</code>.</p>
Examples	To create a boot floppy disk, in the MATLAB window, type xpcbootdisk
See Also	The xPC Target functions <code>setpcenv</code> , <code>getpcenv</code> , <code>updatepcenv</code> , <code>xpcbootdisk</code> , and <code>xpcsetup</code> . See also, the procedures “Creating a Target Boot Disk with Graphical Interface” on page 6-17 and “Creating a Target Boot Disk with Command Line Interface” on page 6-19.

Purpose	Open a scope manager window on the host PC.
Syntax	MATLAB Command Line xpcscope
Description	This graphical user interface (GUI) allows you to define scopes that display on your host PC, choose signals, and control the data acquisition process.
See Also	The xPC Target function <code>xpctgscope</code> and the procedures “Signal Tracing with xPC Target GUI” on page 3-26 and “Signal Tracing with xPC Target GUI (Target Manager)” on page 3-31.

xpcsetup

Purpose	Open the Setup window
Syntax	MATLAB Command Line xpcsetup
Description	This graphical user interface (GUI) allows you to: <ul style="list-style-type: none">• Enter and change environment properties• Create an xPC Target boot floppy disk
See Also	See also the functions setxpcenv, getxpcenv, updatexpcenv, xpcbootdisk, and the procedures “I/O boards - If you use I/O boards on the target PC, you need to correctly install the boards. See the manufactures literature for installation instructions.” on page 2-12, “Environment Properties for Network Communication” on page 2-20, and.

Purpose	Test communication between the host and target computers
Syntax	MATLAB Command Line xpctargetping
Examples	Check for communication between the host PC and target PC. xpctargetping
Description	<p>Ping's the target PC from the host PC and returns either 'success' or 'failed'. If the xPC Target kernel is loaded, running, and communication is working properly, this function returns the value 'success'.</p> <p>This function works with both RS232 and TCP/IP communication.</p> <pre>ans = success</pre>
See Also	The xPC Target procedure "Testing and Troubleshooting the Installation" on page 2-26.

xpctargetspy

Purpose	Open an xPC Target Spy window on the host PC
Syntax	MATLAB Command Line xpctargetspy
Description	<p>This graphical user interface (GUI) allows you to upload displayed data from the target PC.</p> <p>The behavior of this function depends on the value for the environment property TargetScope.</p> <ul style="list-style-type: none">• If TargetScope is enabled, a single graphics screen is uploaded. The screen is not continually updated because of a higher data volume when a target graphics card is in VGA mode. To update the host screen with another target screen, move the pointer into the Spy window and left-click.• If TargetScope is disabled, text output is continuously transferred every second to the host and displayed in the window.
Examples	<p>To open the Target Spy window, in the MATLAB window, type</p> xpctargetspy
See Also	The xPC Target procedures “I/O boards - If you use I/O boards on the target PC, you need to correctly install the boards. See the manufactures literature for installation instructions.” on page 2-12 and “Environment Properties for Network Communication” on page 2-20.

Purpose	Test the xPC Target installation		
Syntax	MATLAB Command Line xpctest xpctest('reboot_flag')		
Arguments	<table><tr><td>reboot_flag</td><td>noreboot. Skips the reboot test. Use this option if the target hardware does not support software rebooting. Value is 'noreboot'</td></tr></table>	reboot_flag	noreboot. Skips the reboot test. Use this option if the target hardware does not support software rebooting. Value is 'noreboot'
reboot_flag	noreboot. Skips the reboot test. Use this option if the target hardware does not support software rebooting. Value is 'noreboot'		
Description	<p>Series of xPC Target tests to check the correct functioning of the following xPC Target tasks:</p> <ul style="list-style-type: none">• Initiate communication between the host and target computers.• Reboot the target PC to reset the target environment.• Build a target application on the host PC.• Download a target application to the target PC.• Check communication between the host and target computers using commands.• Execute a target application.• Comparing the results of a simulation and the target application run. <p>xpctest('noreboot') skips test 2. Use this option if target hardware does not support software rebooting.</p>		
Examples	<p>If the target hardware does not support software rebooting, and to skip test 2, in the MATLAB window, type</p> <pre>xpctest('noreboot')</pre>		
See Also	The procedures “Testing and Troubleshooting the Installation” on page 2-26 and “Test 1, Ping Target System Standard Ping” on page 2-27.		

xpctgscope

Purpose	Open the target scope manager window.
Syntax	MATLAB Command Line xpctgscope
Description	This graphical user interface (GUI) allows you to define scopes that display on your target PC, choose signals, and control the data acquisition process.
See Also	The xPC Target function xpcscope and the procedures “Signal Tracing with xPC Target GUI (Target Manager)” on page 3-31 and “Signal Tracing with xPC Target GUI” on page 3-26.

Purpose	Disconnect the target PC from the current client application
Syntax	MATLAB Command Line <code>xpcwwwenable</code>
Description	Use this function to disconnect the target application from MATLAB before you connect to the Web browser. Also, you can use this function to connect to MATLAB after using a Web browser, or switch to another Web browsers.

Target Object Reference

Target Object	7-3
What is a Target Object?	7-3
Target Object Properties	7-4
Target Object Methods	7-9
Target PC Commands	7-11
 Using Target Objects	 7-13
Displaying Target Object Properties	7-13
Setting the Value of a Target Object Property from the Host PC	7-14
Setting the Value of a Target Object Property from the Target PC	7-15
Getting the Value of a Target Object Property	7-16
Using the Method Syntax with Target Objects	7-17

Use target objects to run and control real-time applications on the target PC.

This chapter includes the following sections:

- **“Target Object”** - Definition, properties, and methods
- **“Using Target Objects”** - Changing properties, and running methods

Target Object

xPC Target uses a target object to represent the target application and target kernel. An understanding of the target object properties and methods will help you to control and test your application on the target PC.

This section includes the following topics:

- **“What is a Target Object?”**
- **“Target Object Properties”**
- **“Target Object Methods”**

What is a Target Object?

A target object on the host PC represents the interface to a target application and the kernel on the target PC. You use target objects to run and control the target application.

When you change a target object property on the host PC, information is exchanged with the target PC and the target application.

To create a target object:

- Build a target application. xPC Target creates a target object during the build process.
- Use the target object constructor function `xpc`. In the MATLAB window, type `tg = xpc`.

A target object has associated properties and methods specific to that object.

Target Object Properties

Target object properties let you access information from your target application and control its execution. You can view and change these properties using target object methods.

The properties for a target object are listed in the following table. This table includes a description of the properties and which properties you can change directly by assigning a value.

Table 7-1: List of Target Object Properties

Property	Description	Write
Connected	Communication status between the host PC and the target PC. Values are 'Yes' or 'No'.	
Appl i cat i on	Name of the Simulink model and target application build from that model.	
Mode	Type of Real-Time Workshop code generation. Values are ' Real - Ti me Si ngl etaski ng' , ' Real - Ti me Mul ti taski ng' , or ' Accel erate' . The default value is ' Real - Ti me Si ngl etaski ng' .	
	Note Even if you select Real-Time Multitasking, the actual mode can be Real-Time Singletasking. This happens if your model contains only one or two tasks and the sample rates are equal.	
Status	Execution status of your target application. Values are ' stopped' or ' runni ng' .	
CPUoverload	CPU status for overload. If the target application requires more CPU time than the sample time of the model, this value is set from ' none' to ' detected' and the current run is stopped. Correcting CPUoverload requires either a faster processor or a larger sample time.	

Table 7-1: List of Target Object Properties

Property	Description	Write
ExecTi me	Execution Time. Time, seconds, since your target application started running. When the target application stops, the total execution time is displayed.	
Sessi onTi me	Time since the kernel started running on your target PC. This is also the elapsed time since you booted the target PC. Values are in seconds.	
StopTi me	Time when the target application stops running. Values are in seconds. The original value is set in the Simulink Simulation Parameters dialog box. When the Executi onTi me reaches the StopTi me, the application stops running.	Yes
Sampl eTi me	Time between samples. This value equals the step size, in seconds, for updating the model equations and post the outputs.	Yes
AvgTET	Average task execution time. This value is an average of the measured CPU times, in seconds, to run the model equations and post outputs during each sample interval. Task execution time is nearly constant with minor deviations due to cache, memory access, interrupt latency, and multirate model execution.	
Mi nTET	Minimum task execution time. Corresponds to the fastest time (smallest time measured), in seconds, to update model equations and post outputs.	

Table 7-1: List of Target Object Properties

Property	Description	Write
MaxTET	Maximum task execution time. Corresponds to the slowest time (longest time measured), in seconds, to update model equations and post outputs.	
ViewMode	Displays either all scopes or a single scope on the target PC. Values are 'all' or a single scope index. This property is active only if the environment property TargetScope is set to enabled.	Yes
TimeLog	Storage in the MATLAB workspace for the time or t-vector logged during execution of the target application.	
StateLog	Storage in the MATLAB workspace for the state or x-vector logged during execution of the target application.	
OutputLog	Storage in the MATLAB workspace for the output, or y-vector logged during execution of the target application.	
TETLog	Storage in the MATLAB workspace for a vector containing task execution times during execution of the target application. To enable logging of the TET, you need to check the Log Task Execution Time box located at Simulation Parameters dialog box> Real-Time Workshop page > Category:xPC Target code generation options group	

Table 7-1: List of Target Object Properties

Property	Description	Write
MaxLogSamples	<p>Maximum number of samples for each logged signal within the circular buffers for TimeLog, StateLog, OutputLog, and TETLog. StateLog and OutputLog can have one or more signals.</p> <p>This value is calculated by dividing the Signal Logging Buffer Size by the number of logged signals. The Signal Logging Buffer Size box is located at Simulation Parameters dialog box> Real-Time Workshop page > Category:xPC Target code generation options group.</p>	
NumLogWraps	The number of times the circular buffer wrapped. The buffer wraps each time the number of samples exceeds MaxLogSamples.	
LogMode	<p>Controls which data points are logged.</p> <ul style="list-style-type: none"> • Equi-distant time. Logs a data point at every time interval. Set value to 'normal'. • Equi-distant amplitude. Logs a data point only when one of the output values from the OutputLog changes by a specified amplitude. Set value to a signal value. 	Yes
Scopes	List of index numbers with one index for each scope.	
NumSignals	The number of signals from your Simulink model that are available to be viewed with a scope.	
ShowSignals	Flag set to view or hide the list of signals from your Simulink blocks. This list is shown when you display the properties for a target object. Values are 'on' or 'off'.	Yes

Table 7-1: List of Target Object Properties

Property	Description	Write
Signal s	<p>List of viewable signals. This list is visible only when ShowSignals is set to ' on' .</p> <ul style="list-style-type: none"> Property name. S0, S1. . . Property value. Value of the signal Block Name. Name of the Simulink block the signal is from. 	
S#	Property name for a signal.	
NumParameters	The number of parameters from your Simulink model that you can tune or change.	
ShowParameters	Flag set to view or hide the list of parameters from your Simulink blocks. This list is shown when you display the properties for a target object. Values are ' on' or ' off' .	Yes
Parameters	<p>List of tunable parameters. This list is visible only when ShowParameters is set to ' on' .</p> <ul style="list-style-type: none"> Property name. P0, P1. . . Property value. Value of the parameter in a Simulink block. Type. Datatype of the parameter. Always double. Size. Size of the the parameter. For example, scalar, 1x2 vector, or 2x3 matrix. Parameter name. Name of a parameter in a Simulink block. Block name. Name of a Simulink block 	Yes
P#	Property name for block parameter.	

Target Object Methods

The target object methods allow you to control a target application on the target PC from the host PC.

Target object methods are entered in the MATLAB window on the host PC. You can also control the target application from the target PC using target PC commands. See “Target PC Commands” on page 7-11.

The methods are listed in the following table.

Table 7-2: List of Target Object Methods

Method	Description
xpc	Creates a target object on the host PC (constructor).
set	Sets writable target object properties to the specified value.
get	Returns the value of readable properties from a target object.
start	Starts the execution of a target application on the target PC.
stop	Stops the execution of a target application on the target PC.
load	Downloads a target application from the host PC to the target PC.
unload	Unloads a target application from the target PC. If a target application is running, it is stopped and unloaded.
addscope	Creates a new scope with type 'host' or 'target' on the target PC.
getscope	Returns the properties of a previously created scope from the target PC. The scope properties can be assigned to a MATLAB variable to create a scope object.

Table 7-2: List of Target Object Methods

Method	Description
remscope	Removes a scope from the target PC. This method does not remove the scope object, on the host PC, that represent the scope.
getparamid	Returns the property name or index of a parameter from the target object.
getsignalid	Returns the property name or index of a signal from the target object.
getlog	Uploads and returns one of the data logs from the target PC to the host PC. TimeLog, StateLog, OutputLog, TETLog
reboot	Reboot the target PC. If a target application is running, the target application is stopped, and then the target PC is rebooted.
close	Close the serial connection to the target PC so that the host PC can use the COMM port for another device.

Target PC Commands

The target PC commands allow you to control a target application on the target PC from the target PC.

Target PC commands are entered in the target PC command window on the target PC. You can also control the target application from the host PC using target object methods. See “Target Object Methods” on page 7-9.

The commands are listed in the following table.

Table 7-3: List of Target PC Commands

Command	Description
del all var	Delete all variables. Syntax: del var
del var	Delete a variable. Syntax: del var <i>variable_name</i>
getpar	Displays the value of a block parameter using the parameter index. Syntax: setpar <i>parameter_index</i>
getvar	Display the value of a variable. Syntax: getvar <i>variable_name</i>
P#	Display or change the value of a block parameter. For example, P2 or P2=1. 23e- 4 Syntax: <i>parameter_name</i> , or <i>parameter_name</i> = <i>floating_point_number</i>
S#	Displays the value of a signal. For example, S2. Syntax: <i>signal_name</i> .
sampl etime	Enter a new sample time. Syntax: sampl etime = <i>floating_point_number</i>

Table 7-3: List of Target PC Commands

Command	Description
setpar	<p>Changes the value of a block parameter using the parameter index.</p> <p>Syntax: <code>setpar parameter_index = floating_point_number</code></p>
setvar	<p>Sets a variable to a value. Later you can use that variable to do a macro expansion.</p> <p>Syntax: <code>setvar variable_name = target_pc_command</code></p> <p>For example, you can type <code>setvar aa=startscope 2</code>, <code>setvar bb=stopscope 2</code></p>
showvar	<p>Display a list of variables.</p> <p>Syntax: <code>showvar</code></p>
stoptime	<p>Enter a new stop time. Use <code>inf</code> to run the target application until you manually stop it or reset the target PC.</p> <p>Syntax: <code>stoptime = floating_point_number</code></p>
viewmode	<p>Zoom in to one scope, or zoom out to all scopes.</p> <p>Syntax: <code>viewmode scope_number</code>, or <code>viewmode 'all'</code></p>

Using Target Objects

xPC Target uses a target object to represent the target application and target kernel. This section shows some of the common tasks that you use with target objects.

This section includes the following topics:

- **“Displaying Target Object Properties”**
- **“Setting the Value of a Target Object Property from the Host PC”**
- **“Setting the Value of a Target Object Property from the Target PC”**
- **“Getting the Value of a Target Object Property”**
- **“Using the Method Syntax with Target Objects”**

Displaying Target Object Properties

You may want to list the target object properties to monitor a target application. The properties include the execution time, and average task execution time.

After you build a target application and target object from a Simulink model, you can list the target object properties. This procedure uses the default target object name `tg` as an example:

- 1 In the MATLAB window, type
`tg`

The current target application properties are uploaded to the host PC, and MATLAB displays a list of the target object properties with the updated values.

Note the target objects properties for TimeLog, StateLog, OutputLog, and TETLog are not updated at this time.

For a list of target object properties with a description, see “Target Object Properties” on page 7-4.

Setting the Value of a Target Object Property from the Host PC

You can change a target object property by using xPC Target methods on the host PC.

With xPC Target you can use either a function syntax or an object property syntax. The syntax `set(target_object, property_name, new_property_value)` can be replaced by:

```
target_object.property_name = new_property_value.
```

For example, to change the stop time mode for the target object `tg`:

- 1 In the MATLAB window, type

```
tg.stoptime = 1000
```

- 2 Alternately, you could type

```
set(tg, 'stoptime', 1000)
```

Parameters are also target object properties. For example, to change the frequency of the signal generator in the model `xpcosc`:

- 1 In the MATLAB window, type

```
tg.p2 = 30
```

- 2 Alternately, you could type

```
set(tg, 'p2', 30)
```

When you change a target object property, the new property value is downloaded to the target PC. The xPC Target kernel then receives the information and changes the behavior of the target application.

To get a list of the writable properties, type `set(target_object)`. The build process assigns the default name of the target object to `tg`.

Note Method names are case-sensitive and need to be complete, but property names are not case-sensitive and need not be complete as long as they are unique.

Setting the Value of a Target Object Property from the Target PC

You can type commands directly from a keyboard on the target PC. These commands create a temporary difference between the behavior of the target application and the properties of the target object. The next time you access the target object, the properties are updated from the target PC:

- 1 On the target PC keyboard, press C, or point the target mouse in the command window.

The target PC activates the command window.

- 2 Type a target command. For example, to change the frequency of the signal generator (parameter 2) in the model `xpcosc`, type

```
setpar 2=30
```

- 3 Change the stop time. For example to set the stop time to 1000 type

```
stoptime = 1000
```

The parameter changes are made to the target application but not to the target object. When you type any xPC Target command in the MATLAB command window, the target PC returns the present properties to the target object.

Note The target PC command `setpar` does not work for vector parameters.

Getting the Value of a Target Object Property

You can list a property value in the MATLAB window, or assign that value to a MATLAB variable. With xPC Target you can use either a function syntax or an object property syntax.

The syntax `get(target_object, property_name)` can be replaced by

`target_object.property_name`

For example, to access the start time:

- 1 In the MATLAB window, type

`endrun = tg.stoptime`

- 2 Alternately, you could type

`endrun = get(tg, 'stoptime')` or `tg.get('stoptime')`

Signals are also target object properties. For example, to get the value of the Integrator1 signal from the model `xpcosc`:

- 1 In the MATLAB window, type

`outputvalue = tg.S0`

- 2 Alternately, you could type

`outputvalue = get(tg, 's2')` or `tg.get('s2')`

To get a list of readable properties, type `target_object`. Without assignment to a variable, the property values are listed in the MATLAB window.

Note Method names are case-sensitive and need to be complete, but property names are not case-sensitive and need not be complete as long as they are unique.

Using the Method Syntax with Target Objects

Use the method syntax to run a target object method. The syntax `method_name(target_object, argument_list)` can be replaced with:

```
target_object.method_name(argument_list)
```

Unlike properties, for which partial but unambiguous names are permitted, method names must be entered in full, and in lowercase. For example, to add a scope of type target with a scope index of 1:

- 1 In the MATLAB window, type

```
tg.addscope('target',1)
```

- 2 Alternately, you could type

```
addscope(tg, 'target', 1)
```

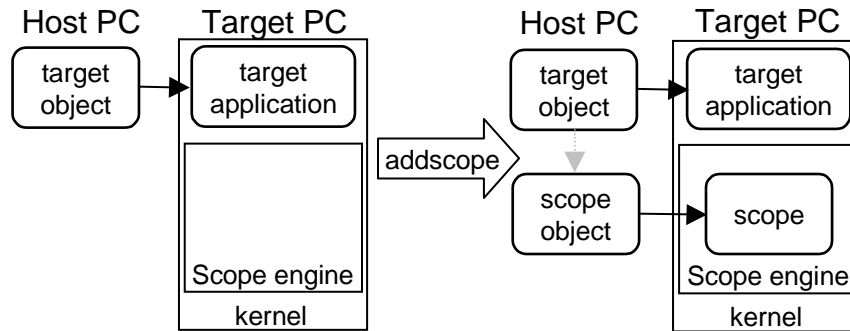
addscope

Purpose	Creates one or more scopes on the target PC	
Syntax	MATLAB command line	
	Creating a scope and scope object without assigning to a MATLAB variable.	
	<pre>addscope(target_object, 'scope_type', new_scope_index) target_object.addscope('scope_type', new_scope_index).</pre>	
	Creating a scope, scope object, and assign to a MATLAB variable.	
	<pre>scope_object = addscope(target_object, 'scope_type', new_scope_index) scope_object = target_object.addscope('target', new_scope_index)</pre>	
Arguments	Target PC command line - When using this command on the target PC, it is limited to adding a scope of type target.	
	<pre>addscope addscope new_scope_index</pre>	
	target_object	Name of a target object.
	scope_type	Values are 'host' or 'target'. This argument is optional with host as the default value.
Description	new_scope_index	Vector of new scope indices. This argument is optional with the next available integer in the target object property Scopes as the default value.
		If you enter a scope index for an existing scope object, the result is an error.
	Method of a target object. Creates a scope on the target PC, a scope object on the host PC, and updates the target object property Scopes. This method returns a scope object vector. If the result is not assigned to a variable, the scope object properties are listed in the MATLAB window. If you try to add a scope with the same index as an existing scope, the result is an error.	
	A scope acquires data from the target application and displays that data on the target PC or uploads the data to the host PC.	

All Scopes of type target or host run on the target PC

Scope of type target - Data collected is displayed on the target screen and acquisition of the next data package is initiated by the kernel.

Scope of type host - Collects data and waits for a command from the host PC for uploading the data. The data is then displayed using the host scope GUI (xpcscope) or other MATLAB functions.



Examples

Create a scope and scope object `sc1` using the method `addscope`. A target scope is created on the target PC with an index of 1, a scope object is created on the host PC, and it is assigned to the variable `sc1`. The target object property `Scopes` is changed from `No scopes defined` to 1.

```
sc1 = addscope(tg, 'target', 1) or sc1 = tg.addscope('target', 1)
```

Create a scope with the method `addscope` and then to create a scope object, corresponding to this scope, using the method `getscope`. A target scope is created on the target PC with an index of 1, a scope object is created on the host PC, but it is not assigned to a variable. The target object property `Scopes` is changed from `No scopes defined` to 1.

```
addscope(tg, 'target', 1) or tg.addscope('target', 1)
sc1 = getscope(tg, 1) or sc1 = tg.getscope(1)
```

Create two scopes using a vector of scope objects `scvector`. Two target scopes are created on the target PC with a scope index of 1 and 2, two scope objects are created on the host PC that represent the scopes on the target PC. The target object property `Scopes` is changed from `No scopes defined` to 1, 2.

```
scvector = addscope(tg, 'target', [1, 2])
```

addscope

See Also

The xPC Target target object methods `remscope`, `getscope`. The xPC Target GUI function `xpcscope`. The xPC target M-file demo scripts listed in “xPC Target Demos” on page 6-21.

Purpose	Closes the serial port connecting the host PC with the target PC	
Syntax	MATLAB command line <code>close(target_object)</code>	
Arguments	<code>target_object</code>	Name of a target object.
Description	Method of a target object. If you want to use the serial port for another function without quitting MATLAB, for example a modem, use this function to close the connection.	

get

Purpose	Return the property values for target and scope objects.				
Syntax	MATLAB command line <code>get(target_object, 'target_object_property')</code>				
Arguments	<table><tr><td><code>target_object</code></td><td>Name of a target object</td></tr><tr><td><code>target_object_property</code></td><td>Name of a target object property.</td></tr></table>	<code>target_object</code>	Name of a target object	<code>target_object_property</code>	Name of a target object property.
<code>target_object</code>	Name of a target object				
<code>target_object_property</code>	Name of a target object property.				
Description	Method of target objects. Gets the value of readable target object properties from a target object.				
Examples	List the value for the target object property <code>StopTime</code> . Notice the property name is a string, in quotes, and not case-sensitive. <code>get(tg, 'stoptime') or tg.get('stoptime')</code> <code>ans = 0.2</code>				
See Also	The xPC Target target object method <code>set</code> . The scope object methods <code>get</code> and <code>set</code> . The built in MATLAB functions <code>get</code> and <code>set</code> .				

Purpose Get all or part of the output logs from the target object

Syntax **MATLAB command line**

```
log = getlog(target_object, 'log_name', start_time,  
number_points, interleave)
```

Arguments

log	User defined MATLAB variable.
log_name	Values are TimeLog, StateLog, OutputLog, or TETLog. This argument is required.
first_point	First data point. The logs begin with 1. This argument is optional. Default is 1
number_points	Number of points after the start time. This argument is optional. Default is all points in log.
interleave	1 returns all sample points. n returns every nth sample point. This argument is optional, Default is 1.

Description

Method of a target object. Use this function instead of the function `get` when you want only part of the data.

Examples

To get the first 1000 points in a log.

```
Outlog = getlog(tg, 'TETLog', 0, 1000)
```

To get every other point in the output log and plot values.

```
Output_log = getlog(tg, TETLog, 0, , 2)  
Time_log = getlog(tg, TimeLog, 0, , 2)  
plot(time_log, output_log)
```

See Also

The xPC Target target object methods `get`. The procedures “Entering the Simulation Parameters” on page 3-8, “Entering the Simulation Parameters” on page 3-8.

getparamid

Purpose Get a parameter index or property name from the parameter list

Syntax **MATLAB command line**

```
getparamid(target_object, 'block_name', 'parameter_name')
getparamid(target_object, 'block_name', 'parameter_name',
'flag')
```

Arguments	target_object	Name of a target object. The default name is tg.
	block_name	Simulink block path and name.
	parameter_name	Name of a parameter within a Simulink block
	flag	If flag = property, then return the property name for the parameter. If flag = numeric, then return a number index. This argument is optional with the default behavior to return a property name.

Description Method of a target object. Returns the index of a parameter in the parameter list based on the path to the parameter name. The names must be entered in full and are case-sensitive.

Examples Get the property name for the parameter Gain in the Simulink block Gain1, incrementally increase gain, and pause to observe signal trace.

```
id = getparamid(tg, 'Subsystem/Gain1', 'Gain')
for i = 1 : 3
    set(tg, id, i*2000);
    pause(1);
end
```

Get the property name (P0, P1, . . .) of a single block.

```
getparamid(tg, 'Gain1', 'Gain')
```

Get the property index (0, 1, . . .) of a single block.

```
getparamid(tg, 'Gain1', 'Gain', 'numeric')
```

P5 is a property of the target object. For example, you could assign a value to the gain with tg.p5 = 1000.

See Also

The xPC Target scope object method `getsignalid`. The xPC target M-file demo scripts listed in “xPC Target Demos” on page 6-21.

Purpose Gets a scope object pointing to a scope already defined in the kernel

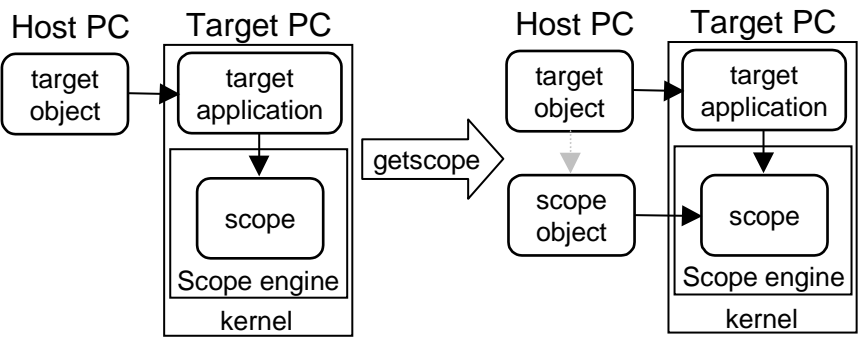
Syntax **MATLAB command line**

```
scope _obj ect_vector = getscope(target_obj ect, scope_i ndex)
```

```
scope_obj ect_vector = target_obj ect. getscope(scope_i ndex)
```

Arguments	target_object	Name or a target object.
	scope_i ndex_vector	Vector of existing scope indices listed in the target property Scopes . The vector may have only one element.
	scope_object_vector	MATLAB variable for a new scope object vector. The vector many have only one scope object.

Description Method of a target object. Returns a scope object vector. If you try to get an nonexistent scope, the result is an error. The list of existing scopes may be retrieved using the method `get(target_object, 'scopes')` or `target_object.scopes`.



Examples If your Simulink model has an xPC Target scope block, a scope of type target is created at the time the target application is downloaded to the target PC. To change the number of samples, you need to create a scope object and then change the scope object property `NumSamples`.

```
sc1 = getscope(tg, 1) or sc1 = tg.getscope(1)
sc1.NumSample = 500
```

To get the properties of all scopes on the target PC and create a vector of scope objects on the host PC. If the target object has more than one scope, creates a vector of scope objects.

```
scvector = getscope(tg)
```

See Also

The xPC Target target object methods `addscope` and `remscope`. The xPC target M-file demo scripts listed in “xPC Target Demos” on page 6-21.

getsignalid

Purpose	Get the signal index or property name from the signal list	
Syntax	MATLAB command line getsignalid(target_object, 'block_name') getsignalid(target_object, 'block_name', 'flag')	
Arguments	target_object	Name of an existing target object.
	block_name	Name of a Simulink block from you model.
	flag	If flag = property, then return the property name for the signal. If flag = numeric, then return a number index. This argument is optional with the default behavior to return a number.
Description	Method of a target object. Returns the index or name of a signal from the signal list, based on the path to the signal name. The block names must be entered in full and are case-sensitive.	
Examples	<p>Get the property name for the parameter Gain in the Simulink block Gain1.</p> <pre>getsignalid(tg, 'Gain1') or tg.getsignal('Gain1') ans = S6</pre> <p>Get the property index for the parameter Gain in the Simulink block Gain1.</p> <pre>getsignalid(tg, 'Gain1', 'Gain', 'numeric') ans = 6</pre> <p>S6 is a property of the target object. For example, you could get the value of a signal with signal_6 = tg.s6.</p>	
See Also	The target object method getparamid. The xPC target M-file demo scripts listed in “xPC Target Demos” on page 6-21.	

Purpose	Download a target application to the target PC				
Syntax	MATLAB command line load(target_obj ect, ' target_appl i cati on') target_obj ect. load(' target_appl i cati on')				
Arguments	<table><tr><td>target_obj ect</td><td>Name of an existing target object</td></tr><tr><td>target_application</td><td>Simulink model and target application name.</td></tr></table>	target_obj ect	Name of an existing target object	target_application	Simulink model and target application name.
target_obj ect	Name of an existing target object				
target_application	Simulink model and target application name.				
Description	<p>Method of a target object. Before using this function, the target PC must be booted with the xPC Target kernel, and the target application must be built in the current working directory on the host PC.</p> <p>If an application was previously loaded, the old target application is first unloaded before downloading the new target application. The method load is called automatically after the RTW build process.</p>				
Examples	<p>Load the target application xpcosc represented by the target object tg.</p> <pre>load(tg, ' xpcosc') or tg. load(' xpcosc') +tg or tg. start or start(tg)</pre>				
See Also	The xPC Target function unl oad. The xPC target M-file demo scripts listed in “xPC Target Demos” on page 6-21.				

reboot

Purpose	Reboot the target PC	
Syntax	MATLAB command line <code>reboot (<i>target_object</i>)</code>	
	Target PC command line <code>reboot</code>	
Arguments	<code>target_obj ect</code>	Name of an existing target object
Description	Method of a target object. Reboots the target PC, and if a target boot disk is still present, the xPC target kernal is reloaded.	
	You can also use this method to reboot the target PC back to Windows after removing the target boot disk.	
	Note This method may not work on some target hardware.	
See Also	The xPC Target target object methods <code>load</code> and <code>unload</code> .	

Purpose Remove a scope from the target PC.

Syntax **MATLAB command line**

```
remscope(target_obj ect, scope_i ndex_vector)
target_obj ect.remscope(scope_i ndex_vector)

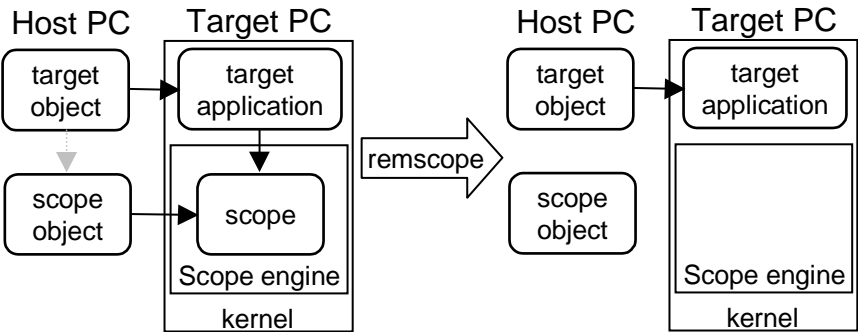
remscope(target_obj ect)
target_obj ect.remscope

Target PC command line
```

```
remscope scope_i ndex
remscope 'all'
```

Arguments	target_obj ect	Name of a target object. The default name it tg.
	scope_i ndex_vector	Vector of existing scope indices listed in the target property Scopes.
	scope_index	Single scope index.

Description Method of a target object. If a scope index is not given, then the method remscope deletes all scopes on the target PC. The method remscope has no return value. The scope object representing the scope on the host PC is not deleted.



Examples Remove a single scope.

```
remscope(tg, 1) or tg.remscope(1)
```

Remove two scopes.

```
remscope(tg, [1 2]) or tg.remscope([1, 2])
```

Remove all scopes.

```
remscope(tg) or tg.remscope
```

See Also

The xPC Target target object methods `addscope` and `getscope`. The xPC target M-file demo scripts listed in “xPC Target Demos” on page 6-21.

Purpose	Change property values for target objects									
Syntax	<p>MATLAB command line</p> <pre>set(target_obj ect)</pre> <pre>set(target_obj ect, property_name1, property_val ue1, property_name2, property_val ue2, . . .)</pre> <pre>target_obj ect. set(' property_name1', property_val ue1)</pre> <pre>set(target_obj ect, property_name_vector, property_val ue_vector)</pre> <pre>target_obj ect_name. property_name = property_val ue</pre> <p>Target PC command line - Commands are limited to the target object properties: stoptime, sampletime, and parameters.</p> <pre>parameter_name = parameter_val ue stoptime = fl oating_poi nt_number sampl etime = fl oating_poi nt_number</pre>									
Arguments	<table><tr><td>target_obj ect</td><td>Name of a target object</td></tr><tr><td>property_name</td><td>Name of a scope object property. Always use quotes</td></tr><tr><td>property_val ue</td><td>Value for a scope object property. Always use quotes for character strings, quotes are optional for numbers.</td></tr><tr><td>parameter_name</td><td>The letter p followed by the parameter index. For example, p0, p1, p2.</td></tr></table>		target_obj ect	Name of a target object	property_name	Name of a scope object property. Always use quotes	property_val ue	Value for a scope object property. Always use quotes for character strings, quotes are optional for numbers.	parameter_name	The letter p followed by the parameter index. For example, p0, p1, p2.
target_obj ect	Name of a target object									
property_name	Name of a scope object property. Always use quotes									
property_val ue	Value for a scope object property. Always use quotes for character strings, quotes are optional for numbers.									
parameter_name	The letter p followed by the parameter index. For example, p0, p1, p2.									
Description	<p>Method of a target object. Sets the properties of the target object. Not all properties are user-writable.</p> <p>Properties must be entered in pairs, or using the alternate syntax, as one-dimensional cell arrays of the same size. This means they have to both be row vectors or both column vectors, and the corresponding values for properties in property_name_vector are stored in property_value_vector.</p> <p>The function set typically does not return a value. However, if called with an explicit return argument, for example, a = set(target_object, property_name,</p>									

property_value), it returns the value of the properties after the indicated settings have been made.

Examples

Get a list of writable properties for a scope object.

```
sc1 = getscope(tg, 1)
set(sc1)
```

```
xPC Target Object:
Writable Properties
```

```
StopTime
SampleTime
ViewMode
LogMode           : [0 | 1]
ShowParameters    : [On | {Off}]
ShowSignals       : [On | {Off}]
```

Change the property showsignals to on.

```
tg.set('showsignals', 'on') or set(tg, 'showsignals', 'on')
```

As an alternative to the method set, use the target object property showsignals. In the MATLAB window, type

```
tg.showsignals = 'on'
```

See Also

The xPC Target target object methods get. The scope object methods get and set. The built in MATLAB functions get and set. The xPC target M-file demo scripts listed in “xPC Target Demos” on page 6-21.

Purpose	Start execution of a target application on a target PC.		
Syntax	MATLAB command line start(target_obj ect) target_obj ect. start +target_obj ect Target PC command line start		
Arguments	<table><tr><td>target_obj ect</td><td>Name of a target object. The default name is tg.</td></tr></table>	target_obj ect	Name of a target object. The default name is tg.
target_obj ect	Name of a target object. The default name is tg.		
Description	Method of both target and scope objects. Starts execution of the target application represented by the target object. Before using this method, the target application must be created and loaded on the target PC. If a target application is running, this command has no effect.		
Examples	Start the target application represented by the target object tg. +tg or tg. start or start(tg)		
See Also	The xPC Target target object methods stop on page 7-36, load on page 7-29, and unload on page 7-37. The scope object method stop on page 8-21.		

stop

Purpose	Stop execution of a target application on a target PC.	
Syntax	MATLAB command line	
	stop(target_obj ect) target_obj ect. stop -target_obj ect	
	Target PC command line	
Arguments	stop	
	target_obj ect	Name of a target object.
Description	Stops execution of the target application represented by the target object. If the target application is stopped, this command has no effect.	
Examples	Stop the target application represented by the target object tg. stop(tg) or tg.stop or -tg	
See Also	The xPC Target target object method start on page 7-35. The scope object methods stop on page 8-21 and start on page 8-19.	

Purpose	Removes the current target application from the target PC.	
Syntax	MATLAB command line <code>unload(target_object)</code> <code>target_object.unload</code>	
Arguments	target_object	Name of a target object that represents a target application.
Description	Method of a target object. The kernel goes into loader mode is ready to download new target application from the host PC.	
Examples	Unload the target application represented by the target object tg. <code>unload(tg)</code> or <code>tg.unload</code>	
See Also	The xPC Target methods load and reboot.	

Purpose	Create a target object representing the target application	
Syntax	MATLAB command line target_obj ect = xpc	
Arguments	target_obj ect	Variable name to reference the target object.
Description	Constructor of a target object. The target object represents the target application and target PC. Changes are made to the target application by making changes to the target object using methods and properties.	
Examples	Before you build a target application, you can check the connection between your host and target computers by creating a target object. tg = xpc xPC Obj ect Connected = Yes Appl i cati on = loader	
See Also	The xPC Target methods get on page 7-22, set on page 7-33.	

Scope Object Reference

Scope Object	8-3
What is a Scope Object?	8-3
Scope Object Properties	8-3
Scope Object Methods	8-6
 Using Scope Objects	 8-8
Displaying Scope Object Properties for a Single Scope	8-8
Displaying Scope Object Properties for All Scopes	8-9
Setting the Value of a Scope Property	8-9
Getting the Value of a Scope Property	8-10
Using the Method Syntax with Scope Objects	8-11

Use scope objects to run and control scopes on the target PC.

This chapter includes the following sections:

- **“Scope Object”** - Definition, properties, and methods
- **“Using Scope Objects”** - Changing properties, and running methods

Scope Object

xPC Target uses scopes and scope objects as an alternative to using Simulink scopes and external mode. Understanding the structure of scope objects will help you to develop a mental model of the xPC Target software environment.

This section includes the following topics:

- **“What is a Scope Object?”** - Definition, and ways to create scope objects
- **“Scope Object Properties”** - List of properties with definitions
- **“Scope Object Methods”** - List of methods with definitions

What is a Scope Object?

A scope object on the host PC represents a scope on the target PC. You use scope objects to observe the signals from your target application during a real-time run or analyze the data after the run is finished.

To create a scope object:

- Add an xPC Target scope block to your Simulink model, build the model to create a scope, and then use the target object method `getscope` to create a scope object.
- Use the target object method `addscope` to create a scope, create a scope object and assign the scope properties to the scope object.

A scope object has associated properties and methods specific to that object.

Scope Object Properties

Scope object properties let you select signals to acquire, set triggering modes, and access signal information from the target application. You can view and change these properties using scope object methods

The properties for a scope object are listed in the following table. This table includes a description of the properties and which properties you can change directly by assigning a value.

Table 8-1: List of Scope Object Properties

Property	Description	Write
Appl i cat i on	Name of the Simulink model associated to this scope object.	
ScopeId	A numeric index unique for each scope.	
Status	Indicates whether data is being acquired, the scope is waiting for a trigger, the scope has been stopped (interrupted), or acquisition is finished. Values are ' Acqui ring' , ' Ready for being Triggered' , ' Interrupted' , and ' Fi ni shed' .	
Type	Determines whether the scope is displayed on the host computer or on the target computer. Values are ' host' and ' target' .	
NumSampl es	Number of contiguous samples captured during the acquisition of a data package.	Yes
NumPrePostSampl es	Number of samples collected before or after a trigger event. The default value is 0. Entering a negative value collects samples before the trigger event. Entering a positive value collects samples after the trigger event. If you set TriggerMode to FreeRun, this property has no effect on data acquisition.	

Table 8-1: List of Scope Object Properties

Property	Description	Write
Decimation	<p>A number n, where every nth sample is acquired in a scope window.</p> <p>Note This value is the same as Interleave in a scope window.</p>	Yes
TriggerMode	Trigger mode for a scope. Valid values are 'FreeRun' (default), 'Software', 'Signal', and 'Scope'.	Yes
TriggerSignal	If TriggerMode='Signal', identifies which block output signal to use for triggering the scope. You identify the signal with a signal index from the target object property Signal.	Yes
TriggerLevel	If TriggerMode='Signal', indicates the value the signal has to cross to trigger the scope and start acquiring data. The trigger level can be crossed with either a rising or falling signal.	Yes
TriggerSlope	If TriggerMode='Signal', indicates whether the trigger in on a rising or falling signal. Values are 'Either' (default), 'Rising', or 'Falling'.	Yes
TriggerScope	If TriggerMode='Scope', identifies which scope to use for a trigger. A scope can be set to trigger when another scope is triggered. This is done by setting the slave scope property TriggerScope to the scope index of the master scope.	Yes
Mode	Indicates how a scope displays the signals. Values are 'Numerical', 'Redraw' (default), 'Sliding', or 'Rolling'.	Yes

Table 8-1: List of Scope Object Properties

Property	Description	Write
YLi mi t	Minimum and maximum y-axis values. This property can be set to ' auto ' .	Yes
Grid	Values are ' on ' or ' off ' .	Yes
StartTi me	Time within the total execution time, when a scope begins acquiring a data package.	
Time	Contains the time data for a single data package from a scope.	
Data	Contains the output data for a single data package from a scope.	
Si gnal s	List of signal indices from the target object to display on the scope.	

Scope Object Methods

The scope object methods allow you to control scopes on your target PC. The methods are listed in the following table.

Table 8-2: List of Scope Object Methods

Scope Method	Description
set	Sets writable scope object properties to the specified value. For a list of writable values, use <code>set (scope_obj ect)</code>
get	Returns the value of readable properties from a scope object.
addsi gnal	Adds a signal to a scope and a scope object. The signal is specified using the signal indices from the target object.

Table 8-2: List of Scope Object Methods

Scope Method	Description
remsignal	Removes a signal from a scope and a scope object. The signal is specified using signal indices from the scope object.
start	Starts a scope, but does not necessarily start the acquisition of data. The acquisition of data is dependent on the trigger mode.
stop	Stops a scope and the acquisition of a data.
trigger	If TriggerMode=' Software' , starts the acquisition of data from the target application.

Using Scope Objects

xPC Target uses scope objects to represent scopes on the target PC. This section shows some of the common tasks that you use with scope objects.

This section includes the following topics:

- “**Displaying Scope Object Properties for a Single Scope**”
- “**Displaying Scope Object Properties for All Scopes**”
- “**Setting the Value of a Scope Property**”
- “**Getting the Value of a Scope Property**”
- “**Using the Method Syntax with Scope Objects**”

Displaying Scope Object Properties for a Single Scope

To list the properties of a single scope object `sc1`:

- 1 In the MATLAB window, type

```
sc1 = getscope(tg, 1) or sc1 = tg.getscopes(1)
```

MATLAB creates the scope object `sc1` from a previously created scope.

- 2 Type

```
sc1
```

The current scope properties are uploaded to the host PC, and then MATLAB displays a list of the scope object properties with the updated values. Because `sc1` is a vector with a single element, you could also type, `sc1(1)` or `sc1([1])`.

Note Only scopes with type `host` store data in the properties `scope_object.Time` and `scope_object.Data`.

For a list of target object properties with a description, see “Target Object Properties” on page 7-4.

Displaying Scope Object Properties for All Scopes

To list the properties of all scope objects associated with the target object `tg`:

- 1 In the MATLAB window, type
`getscope(tg)` or `tg.getscope`

MATLAB displays a list of all scope objects associated with the target object.

- 2 Alternately, type
`allscopes = getscope(tg)` or `allscopes = tg.getscope`
`allscopes`

The current scope properties are uploaded to the host PC, and then MATLAB displays a list of all the scope object properties with the updated values. To list some of the scopes, use the vector index. For example, to list the first and third scopes, type `allscopes([1, 3])`.

For a list of target object properties with a description, see “Target Object Properties” on page 7-4.

Setting the Value of a Scope Property

With xPC Target you can use either a function syntax or an object property syntax. The syntax `set(scope_object, property_name, new_property_value)` can be replaced by:

- `scope_object_vector.property_name = new_property_value.`
- `scope_object(index_vector).property_name = new_property_value.`

For example to change the trigger mode for the scope object `sc1`:

- 1 In the MATLAB window, type
`sc1.triggermode = 'signal'`
- 2 Alternately, you could type
`set(sc1,'triggermode','signal')` or `sc1.set('triggermode','signal')`

Assignment for may also be done for a vector of scope objects, for example `allscopes([1, 2]).numsamples = 500`. Notice, the indices are MATLAB vector indices and not xPC Target scope indices.

To get a list of the writable properties, type `set(scope_object)`.

Note Method names are case-sensitive, but property names are not.

Getting the Value of a Scope Property

You can list a property value in the MATLAB window, or assign that value to a MATLAB variable. With xPC Target you can use either a function syntax or an object property syntax.

The syntax `get(scope_object_vector, property_name)` can be replaced by

- `scope_object_vector.property_name`
- `scope_object_vector(index_vector).property_name`

For example to assign the start time from the scope object `sc1`:

1 In the MATLAB window, type

```
beginrun = sc1.starttime
```

2 Alternately, you could type

```
beginrun = get(sc1,'starttime') or sc1.get('starttime')
```

Assignment may also be done using a vector of scope objects, for example `scopetypes = allscopes([1, 2]).type`. Notice, the indices are MATLAB vector indices and not xPC Target scope indices.

To get a list of readable properties, type `scope_object`. Without assigning to a variable, the property values are listed in the MATLAB window.

Note Method names are case-sensitive, but property name are not.

Using the Method Syntax with Scope Objects

Use the method syntax to run a scope object method. The syntax `method_name(scope_object_vector, argument_list)` can be replaced with:

- `scope_object.method_name(argument_list)`
- `scope_object_vector(index_vector).method_name(list of arguments)`

Unlike properties, for which partial but unambiguous names are permitted, method names must be entered in full, and in lowercase. For example, to add signals to the first scope in a vector of all scopes:

- 1 In the MATLAB window, type
`allscopes(1).addsignal([0,1])`
- 2 Alternately, you could type
`addsignal(allscopes(1), [0,1])`

</

addsignal

```
set(sc1, 'Signals', [0, 1]) or sc1.set('signals', [0, 1],
```

Or to directly assign signal values to the scope object property `Signals`.

```
sc1.signals = [0, 1].
```

See Also

The xPC Target scope object methods `remsignal` and `set`. The target object method `addscope` and `getsignalid`.

Purpose	Return the property values for scope objects							
Syntax	MATLAB command line get (scope_obj ect_vector) get (scope_obj ect_vector, ' scope_obj ect_property') get (scope_obj ect_vector, scope_obj ect_property_vector)							
Arguments	<table><tr><td>target_obj ect</td><td>Name of a target object</td></tr><tr><td>scope_obj ect_vector</td><td>Name of a single scope, or name of a vector of scope objects</td></tr><tr><td>scope_object_property</td><td>Name of a scope object property</td></tr></table>		target_obj ect	Name of a target object	scope_obj ect_vector	Name of a single scope, or name of a vector of scope objects	scope_object_property	Name of a scope object property
target_obj ect	Name of a target object							
scope_obj ect_vector	Name of a single scope, or name of a vector of scope objects							
scope_object_property	Name of a scope object property							
Description	Method of scope objects. Gets the value of readable scope object properties from a scope object or the same property from each scope object in a vector of scope objects.							
Examples	<p>List all of the readable properties, along with their present values. This is given in the form of a structure, whose fieldnames are the property names and field values are property values.</p> <pre>get (sc)</pre> <p>List the value for the scope object property Type. Notice the property name is a string, in quotes, and is not case-sensitive.</p> <pre>get (sc, ' type') ans = Target</pre>							
See Also	The xPC Target scope object method set. The target object methods get and set. The built in MATLAB functions get and set.							

remsignal

Purpose	Remove signals from a scope represented by a scope object	
Syntax	MATLAB command line remsignal (scope_obj ect) remsignal (scope_obj ect, si gnal_i ndex_vector) scope_obj ect.remsignal (si gnal_i ndex_vector) Target command line remsignal scope_index = signal_index, signal_index, . . .	
Arguments	scope_obj ect	MATLAB object created with the target object methods addscope or getscope.
	si gnal_i ndex_vector	Index numbers from the scope object property Si gnal s. This argument is optional and if left out all signals are removed.
	si gnal_i ndex	Single signal index.
Description	Method for a scope object. The signals must be specified by their index, which may be retrieved using the target object method getsi gnal i d. If the scope_object_vector has two or more scope object, the same signals are removed from each scope. The argument SIGNALS is optional; if left out, all signals are removed.	
Examples	Remove signals 0 and 1 from the scope represented by the scope object sc1. sc1.get(' si gnal s') ans= 0 1	
	Removed signals from the scope on the target PC with the scope object property Signals updated. remsignal (sc1, [0, 1]) or sc1.remsignal ([0, 1])	
See Also	The xPC Target scope object method remsignal , and the target object method getsi gnal i d.	

Purpose	Change property values for scope objects	
Syntax	MATLAB command line set(scope_object_vector) set(scope_object_vector, property_name1, property_value1, property_name2, property_value2, . . .) scope_object_vector.set('property_name1', property_value1, . . .) set(scope_object, 'property_name', property_value, . . .)	
Arguments	scope_object	Name of a scope object, or a vector of scope objects
	property_name	Name of a scope object property. Always use quotes
	property_value	Value for a scope object property. Always use quotes for character strings, quotes are optional for numbers.
Description	<p>Method for scope objects. Sets the properties of the scope object. Not all properties are user-writable</p> <p>Properties must be entered in pairs, or using the alternate syntax, as one-dimensional cell arrays of the same size. This means they have to both be row vectors or both column vectors, and the corresponding values for properties in property_name_vector are stored in property_value_vector.</p> <p>The function set typically does not return a value. However, if called with an explicit return argument, for example, a = set(target_object, property_name, property_value), it returns the value of the properties after the indicated settings have been made.</p>	
Examples	<p>Get a list of writable properties for a scope object.</p> <pre>sc1 = getscope(tg, 1) set(sc1)</pre> <p>xPC Scope Object: Writable Properties</p>	

```
NumSamples
Decimation
TriggerMode : [{FreeRun} | Software | Signal | Scope]
TriggerSignal
TriggerLevel
TriggerSlope : [{Either} | Rising | Falling]
TriggerScope
Signals
Mode : [Numerical | {Redraw} | Sliding | Rolling]
YLimit
Grid
```

The property value for the scope object `sc1` is changed to on

```
sc1.set('grid', 'on') or set(sc1, 'grid', 'on')
```

See Also

The xPC Target scope object method `get`. The target object methods `set` and `get`. The built in MATLAB functions `get` and `set`.

Purpose	Start execution of a scope on a target PC	
Syntax	MATLAB command line start(scope_object_vector) scope_object_vector.start +scope_object_vector start(getscope(target_object, scope_index_vector)) Target PC command line startscope scope_index startscope 'all'	
Arguments	target_object	Name of a target object.
	scope_object_vector	Name of a single scope object, name of vector of scope objects, list of scope object names in a vector form [scope_object1, scope_object2], or the target object method getscope which returns a scope_object vector.
	scope_index_vector	Index for a single scope, or list of scope indices in vector form.
	scope_index	Single scope index.
Description	Method for scope objects. Starts a scope on the target PC represented by a scope object on the host PC. This method does not necessarily start data acquisition which depends on the trigger settings. Before using this method, a scope must be created. To create a scope, use the target object method addscope or add xPC Target scope blocks to your Simulink model.	
Examples	Start one scope with the scope object sc1. sc1 = getscope(tg, 1) or sc1 = tg.getscope(1) start(sc1) or sc1.start or +sc1 or type start(getscope(tg, 1)) Start two scopes.	

start

```
somescope = getscope(tg, [1, 2]) or somescopes =  
tg.getscope([1, 2])  
start(somescope) or somescopes.start
```

or type

```
sc1 = getscope(tg, 1) or sc1 =tg.getscope(1)  
sc2 = getscope(tg, 2) or sc2 = tg.getscope(2)  
start([sc1, sc2])
```

or type

```
start(getscope(tg, [1, 2]))
```

Start all scopes

```
allscopes = getscope(tg) or allscopes = tg.getscope  
start(allscopes) or allscopes.start or +allscopes
```

or type

```
start(getscope(tg)) or start(tg.getscope)
```

See Also

The xPC Target target object methods `getscope` and `stop`. The scope object method `stop`.

Purpose	Stop execution of a scope on the target PC.	
Syntax	MATLAB command line stop(scope_obj ect_vector) scope_obj ect. stop -scope_obj ect stop(getscope(target_obj ect, scope_i ndex_vector)) Target PC command line stopscope scope_i ndex stopscope 'all'	
Arguments	target_obj ect	Name of a target object.
	scope_obj ect_vector	Name of a single scope object, name of vector of scope objects, list of scope object names in a vector form [scope_object1, scope_object2], or the target object method getscope which returns a scope_object vector.
	scope_i ndex_vector	Index for a single scope, or list of scope indices in vector form.
	scope_i ndex	Single scope index.
Description	Method for a scope objects. Stops the scopes represented by the scope objects.	
Examples	Stop one scope represented by the scope object sc1. stop(sc1) or sc1.stop or -sc1 Stop all scopes with a scope object vector allscopes created with the command allscopes = getscope(tg) or allscopes = tg.getscope. stop(allscopes) or allscopes.stop or -allscopes or type stop(getscope(tg)) or stop(tg.getscope)	

stop

See Also

The xPC Target target object methods `getscope`, `stop`, and `start`. The scope object method `start`.

Purpose	Software trigger the start of data acquisition for one or more scopes.	
Syntax	MATLAB command line <code>trigger(scope_object_vector) or scope_object_vector.trigger</code>	
Arguments	<code>scope_object_vector</code>	Name of a single scope object, name of a vector of scope objects, list of scope object names in a vector form <code>[scope_object1, scope_object2]</code> , or the target object method <code>getscope</code> which returns a <code>scope_object</code> vector.
Description	Method for a scope object. If the scope object property <code>TriggerMode</code> has a value of 'software', then this function triggers the scope represented by the scope object to acquire the number of data points in the scope object property <code>NumSamples</code> . Note Only scopes with type <code>host</code> store data in the properties <code>scope_object.Time</code> and <code>scope_object.Data</code> .	
Examples	Set a single scope to software trigger, trigger the acquisition or one set of samples, and plot data. <pre> sc1 = tg.addscope('host', 1) or sc1=addscope(tg, 'host', 1) sc1.triggermode = 'software' tg.start, or start(tg), or +tg sc1.start or start(sc1) or +sc1 sc1.trigger or trigger(sc1) plot(sc1.time, sc1.data) sc1.stop or stop(sc1) or -sc1 tg.stop or stop(tg) or -tg1 </pre> Set all scopes to software trigger and trigger to start. <pre> allscopes = tg.getscopes allscopes.triggermode = 'software' allscopes.start or start(allscopes) or +allscopes allscopes.trigger or trigger(allscopes) </pre>	

trigger

A

adding

- scope blocks 4-14

advanced tutorial 4-1

advantages

- network communication 2-15
- serial communication 2-12

B

before you install

- obtaining a valid license 2-7
- overview 2-7

block library

- in Simulink 4-7
- with xPC Target 4-4

block parameters

- defining 4-10
- defining scope 4-16

boot disk, see target boot disk

booting the target PC 3-7

build process xiv

- target application 3-13
- troubleshooting 3-15

C

C compiler

- required product xiv

CD-ROM

- installing from 2-8

changing

- environment properties 5-16

changing parameters

- using commands 3-38
- using target object properties 3-38
- xPC Target commands 3-38

changing properties

- environment properties 5-14

command line interface 1-20

- scope object 7-3

- target object 6-3

commands

- xPC Target 5-20

communication

- between computers 1-12

compiler

- required xiv

computer

- communication 1-12
- desktop PC 1-8, 1-9
- host PC 1-8
- industrial PC 1-9
- notebook PC 1-8
- target PC 1-8

connecting

- computers 1-9
- I/O boards 1-11
- real-world 1-11

contacting The MathWorks

- for technical support 2-31
- for valid license 2-7

conventions

- in this guide xiv
- typographical xxi

creating

- boot disk 5-19
- model with I/O blocks 4-3
- model with scope blocks 4-13
- scope objects 3-26, 3-31
- target application 3-7
- target boot disk 2-23, 5-17, 5-19
- target object 3-13

D

defining

- I/O block parameters 4-10
- scope block parameters 4-16

desktop PC 1-8, 1-9

directories

- installed 2-10
- working 2-10
- xpc 2-10
- xpcdemo 2-10

disk

- boot, see target boot disk

diskette, see target boot disk

documentation

- notational conventions xix
- terminology conventions xix

downloadable file

- installation 2-8

downloading

- target application 3-13

E

entering

- environment properties 5-14
- Simulation Parameters 3-8

environment

- network communication 2-20
- serial communication 2-13

environment properties

- changing 5-14, 5-16
- list 5-12
- loading 5-13
- saving 5-13
- updating 5-14, 5-16

Ethernet card

- ISA-bus 2-18

PCI-bus 2-18

expected background xvii

external mode

- Simulink 1-22

F

features

- xPC Target 1-4

files

- installed 2-10
- project directory 2-10
- working directory 2-10
- xpc directory 2-10
- xpcdemos directory 2-10

floppy disk, see target boot disk

functions

- changing parameters 3-38

G

getting

- list of environment properties 5-12
- list of scope object properties 3-36
- list of target object properties 3-36, 3-38

graphical user interface (GUI)

- interaction with 1-19

H

hardware environment 1-8

host computer, see host PC

host PC 1-8

- communication 1-12
- connecting 1-9
- hardware 2-12
- requirements 2-3

I**I/O block library**

- access in Simulink 4-7
- access in xPC Target 4-4

I/O block parameters

- defining 4-10

I/O blocks

- in Simulink model 4-3

industrial PC 1-9**installing**

- Ethernet card for ISA 2-18
- ethernet card for PCI 2-18
- from CD-ROM 2-8
- from downloadable file 2-8
- hardware 2-12
- network communication 2-15
- serial communication 2-12
- testing 2-26

interaction

- command line interface 1-20
- graphical user interface 1-19

ISA-bus

- Ethernet card 2-18

L**license**

- obtaining 2-7

list

- environment properties 5-12
- scope properties 7-3
- target properties 6-4

loading

- environment properties 5-13
- Simulink model 3-3

M**MATLAB**

- required product xii

methods

- scope object 7-6
- target object 6-9, 6-11

N**network communication**

- advantages 2-15
- environment 2-20
- hardware 2-16
- host PC 2-16
- installing 2-15
- setting up 2-15, 2-20
- target PC 2-16

notebook PC 1-8**O****organization of this document xviii****overview**

- command line interface 1-20
- graphical user interface 1-19

P**parameters**

- changing with commands 3-38
- defining block 4-10
- defining scope blocks 4-16

PCI-bus

- Ethernet card 2-18

- properties
 - changing environment 5-16
 - environment list 5-12
 - scope object 7-3
 - target object 6-4
 - updating environment 5-16

R

- Real-Time Workshop
 - required product xiv
- required products
 - C language compiler xiv
 - MATLAB xii
 - Real-Time Workshop xiv
 - Simulink xiii
 - xPC Target xii
- requirements
 - host PC 2-3
 - system 2-3
 - target PC 2-4
- running a simulation
 - on the host PC 3-3
- running target application 3-16

S

- saving
 - environment properties 5-13
- scope blocks
 - adding to model 4-14
 - defining parameters 4-16
 - in Simulink model 4-13
- scope object
 - command line interface 7-3
 - commands 7-3
 - getting list of properties 3-36

- methods 7-6
- methods, see commands
- properties 7-3
- scope objects
 - creating 3-26, 3-31
- selecting
 - signals for tracing 3-26, 3-31
- serial communication
 - advantages 2-12
 - environment 2-13
 - hardware 2-12
 - installing 2-12
 - setting up 2-12
- setting
 - initial working directory 2-11
- Setup window
 - using 5-12
- signal 3-31
- signal tracing
 - selecting signals 3-26
- signals for tracing
 - selecting 3-26, 3-31
- Simulation Parameters
 - entering 3-8
- Simulink
 - external mode 1-22
 - I/O block library 4-7
 - loading a model 3-3
 - required products xiii
- Simulink model
 - adding scope blocks 4-14
 - with I/O blocks 4-3
 - with scope blocks 4-13
- software environment 1-12
- starting
 - target application 3-17

- stopping
 - target application 3-17
- system requirements 2-3
 - host PC 2-3

T

- target application
 - building 3-13
 - creating 3-7
 - downloading 3-13
 - running 3-16
 - starting 3-17
 - stopping 3-17
- target boot disk
 - creating 2-23, 5-17, 5-19
 - with desktop PC 1-9
 - with industrial PC 1-9
- target computer, see target PC
- target object
 - changing parameters 3-38
 - command line interface 6-3
 - commands 6-3
 - getting list of properties 3-36, 3-38
 - methods 6-9, 6-11
 - methods, see commands
 - properties 6-3, 6-4
- target PC 1-8
 - booting 3-7
 - communication 1-12
 - connecting 1-9
 - creating boot disk 5-19
 - hardware 2-12
 - requirements 2-4
 - running target application 3-16
- testing
 - installation 2-26

- troubleshooting
 - build process 3-15
- tutorial
 - advanced 4-1
 - basic 3-1
 - creating a target application 3-7
 - running a simulation 3-3
 - running target application 3-16

U

- updating
 - environment properties 5-14, 5-16
- using
 - setup window 5-12
 - xPC Target commands 5-20
 - xPC Target setup window 5-12
- using this guide
 - conventions xix
 - organization xviii

V

- valid license
 - obtaining 2-7

W

- working directory
 - initial 2-11
 - setting initial 2-11

X**xPC Target**

- commands 5-20

- features 1-4

- interaction 1-18

- overview 1-3

- required products xii

- Setup window 5-12

- what is it? 1-3