xPC Target

For Use with Real-Time Workshop®

- Modeling
- Simulation
- Implementation

Getting Started

Version 2

How to Contact The MathWorks:

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup

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

508-647-7000 Phone

508-647-7001 Fax

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

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

Getting Started with xPC Target

Printing History:

November 2000	First printing	New for Version 1.1 (Release 12)
June 2001	Online only	Revised for Version 1.2 (Release 12.1)
September 2001	Online only	Revised for Version 1.3 (Release 12.1+)
July 2002	Second printing	Revised for Version 2 (Release 13)
September 2003	Online only	Revised for Version 2.0.1 (Release 13 SP1)
October 2004	Online only	Revised for Version 2.0.3 (Release 13SP2)

Contents

Preface

1 Introduction

2 Installation and Configuration

3 Basic Tutorial

4 Signals and Parameters

Index

Preface

xPC Target is part of a family of software products that you use to create real-time systems. Some of these products are required while others you use for special applications. This chapter includes the following sections:

What Is xPC Target?

xPC Target is a solution for prototyping, testing, and deploying real-time systems using standard PC hardware. It is an environment that uses a target PC, separate from a host PC, for running real-time applications.

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

xPC Target lets you to add I/O blocks to your model, and then use the host PC with Real-Time Workshop®, Stateflow Coder (optional) and a C/C++ compiler to create executable code. The executable code is downloaded from the host PC to the target PC running the xPC Target real-time kernel. After downloading the executable code, you can run and test your target application in real time:

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. However, the target PC can be a desktop PC, industrial PC, PC/104, PC/104+, or CompactPCI computer.

Special software requirements — The xPC Target software requires either a Microsoft Visual C/C++ compiler (version 5.0, 6.0, or 7.0) 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 license from The MathWorks. With this additional license, 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.
- Create stand-alone applications for the target PC independent from the host PC that can boot from a floppy drive or an alternate device.
- Create stand-alone GUI applications running on the host PC to control, change parameters, and acquire signal data from a target application. This feature uses the xPC Target API with any programming environment, or the xPC Target COM API with any programming environment, such as Visual Basic, that can use COM objects.

Required Products

xPC Target is a PC-compatible product that you install on a host computer running a Microsoft Windows operating system. xPC Target requires the following products from The MathWorks:

- **MATLAB** — Control and interact with the xPC Target software environment and target application using a command-line interface
- **Simulink** — Model dynamic physical systems and controllers using block diagrams
- **Real-Time Workshop** — Convert Simulink blocks and Stateflow charts into C code
- **C Compiler** — Use a third-party C compiler and Real-Time Workshop to build a target application
- **xPC Target Embedded Option** — Deploy stand-alone target applications and custom GUI applications that communicate with the target application

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 xPC Target functions and the command-line interface or M-file scripts. You use the xPC Target 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 2.0.3 of xPC Target requires the MATLAB on the Release 13 SP2 CD.

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

Simulink

Simulink provides an environment where you model your dynamic 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 replace the model of your physical system with I/O driver blocks connected to the actual physical system, or you can replace the model of your controller with the actual controller. The xPC Target I/O library supports more than 400 driver blocks. As additional drivers become available, you can download updates from the MathWorks Web site at

```
http://www.mathworks.com/support/author/xpc/index.shtml
```

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

Note Version 2.0.3 of xPC Target requires the MATLAB on the Release 13 SP2 CD.

Simulink documentation — For information on using Simulink, see the Using Simulink documentation. 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/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 2.0.3 of xPC Target requires the MATLAB on the Release 13 SP2 CD.

Real-Time Workshop documentation — For information on code generation, see the Real-Time Workshop documentation.

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. xPC Target uses this executable code to create an executable image (target application) that runs with the xPC Target kernel on the target computer.

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 2.0.3 of xPC Target requires Microsoft Visual Studio C/C++ Version 5.0, 6.0, or 7.0. If your are going to use the xPC Target Embedded Option with the COM API, you need to use Visual C/C++ to build your target application and create the model-specific COM library.

Watcom C/C++

Version 2.0.3 of xPC Target requires Watcom C/C++ Version 10.6 or 11.0.

xPC Target Embedded Option

The xPC Target Embedded Option is a separate product that requires an additional license from The MathWorks. You do not need this product for rapid prototyping, but with this additional license, you can

- Boot the target PC from an alternate device other than a floppy disk drive such as a hard disk drive or flash memory.
- Create stand-alone applications for the target PC independent from the host PC that can boot from a floppy drive or an alternate device.
- Create stand-alone GUI applications running on the host PC to control, change parameters, and acquire signal data from a target application. This feature uses the xPC Target API with any programming environment, or the xPC Target COM API with any programming environment, such as Visual Basic, that can use COM objects.

Note Version 2.0.3 of xPC Target requires xPC Target Embedded Option on the Release 13 SP2 CD.

---

xPC Target Embedded Option documentation — Information about the xPC Target Embedded Option is included with the xPC Target documentation.

Related Products

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

For more information about any of these products, see either

- The online documentation for that product if it is installed or if you are reading the documentation from the CD
- The MathWorks Web site, at `http://www.mathworks.com`; see the "products" section

Note The toolboxes listed below all include functions that extend the capabilities of MATLAB. The blocksets all include blocks that extend the capabilities of Simulink.

Product	Description
DSP Blockset	Design and simulate DSP systems
SimMechanics	Model and simulate mechanical systems
Stateflow	Design and simulate event-driven systems
Stateflow Coder	Generate C code from Stateflow charts
Virtual Reality Toolbox	Create and manipulate virtual reality scenes from within MATLAB and Simulink

Documentation and Help

We ship the xPC Target software with Getting Started documentation. This guide and the remaining documentation are available online through the MATLAB Help browser window, or as PDF files that you can view online or print. This section includes the following topics:

- **Installing Online Documentation** — Install HTML files from the Documentation CD or from a Web download
- **Viewing Online Documentation** — View HTML files from your hard drive, the Documentation CD, or the MathWorks Web site
- **Printing the Documentation** — Locate and print PDF files on the Documentation CD or the MathWorks Web site
- **Product and Product News Pages** — Information from the developers for xPC Target

Installing Online Documentation

Installing the online documentation is part of the usual MathWorks install process:

- **Documentation from a CD** — Start the MathWorks installer, and when prompted, select the **Product** and **Documentation** check boxes. Later, during the install process, you will be asked to insert the Documentation CD.
- **Documentation from a Web download** — If you update xPC Target using a Web download, and you want to view the documentation with the MathWorks Help browser, you must install the documentation on your hard drive. Start the Web installer, and as before, select the **Product** and **Documentation** check boxes.

Note During the usual installation of xPC Target from a CD or a Web download, the PDF files for the documentation are not copied to your hard drive. When you select to install the documentation, only the HTML files are copied.

Viewing Online Documentation

You can access the online documentation from HTML files you install on your hard drive, from the Documentation CD, or through the MathWorks technical support Web pages.

To Access HTML Documentation on Your Hard Drive or the Documentation CD

1. In the MATLAB window, and from the **Help** menu, click **Full Product Family Help**.

   The Help browser window opens.

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

   In the right pane, the Help browser displays the xPC Target Roadmap page. If you did not install the HTML help files on your hard drive, a message box opens asking you to insert the Documentation CD.

3. Under the section titled "Learning About xPC Target," select "What's New," or one of the books in the xPC Target documentation set.

   The Help browser displays the overview page for the book you choose.

Note If you installed xPC Target from a Web download, and you choose not to install the HTML help files, the current documentation is neither on your hard drive, nor on the Documentation CD. You need to use the MathWorks technical support Web site for the updated documentation.

To Access HTML Documentation from MathWorks Technical Support

Alternatively, you can view the documentation from the MathWorks technical support Web site. The Web pages are identical to the latest release whether the release was distributed from a CD or a Web download.

1. Open a Web browser.
2. In the address box, enter

   ```
   http://www.mathworks.com/access/helpdesk/help/toolbox/xpc/
   xpc.shtml
   ```

3. Under the section titled "Learning About xPC Target," select "What's New," or one of the books in the xPC Target documentation set.

Printing the Documentation

The documentation for xPC Target is available as PDF files. You need to install Adobe Acrobat Reader 4.0 or later to open and read these files. To download a free copy of Acrobat Reader, see

```
http://www.adobe.com/products/acrobat/main.html
```

To Access PDF Documentation on the Documentation CD

1. Insert the documentation CD into your CD-ROM drive.
2. In the MATLAB window, and from the **Help** menu, click **Full Product Family Help**.

   The MathWorks Help browser window opens.
3. In the left pane, click **xPC Target**.

   In the right pane, the Help browser displays the xPC Target Roadmap page.
4. Under the section titled "Printing the Documentation," select the PDF file you want to print.

Note If you installed xPC Target from a Web download, the current PDF documentation is not available on your hard drive, nor is it available on the Documentation CD. You need to get the updated PDF files from the MathWorks technical support Web site.

To Access PDF Documentation from MathWorks Technical Support

1 Open a Web browser. In the address box, enter

```
http://www.mathworks.com/support/product/XP/productnews/
productnews.shtml
```

2 Under the section titled "Printing the Documentation," select the PDF file you want to print.

Locating PDF Files Directly on the Documentation CD

The following manuals are available on the Documentation CD as PDF files:

- **xPC Target Getting Started** — Located at `Z:\help\pdf_doc\xpc\xpc_target_gs.pdf`
- **xPC Target User's Guide** — Located at `Z:\help\pdf_doc\xpc\xpc_target_ug.pdf`
- **xPC Target I/O Reference Guide** — Located at `Z:\help\pdf_doc\xpc\xpc_target_io_ref.pdf`

Product and Product News Pages

The developers for xPC Target maintain a Product News page at the MathWorks Web site. This is where you will find driver updates, beta release of some new features, and preliminary documentation.

```
http://www.mathworks.com/support/author/xpc/
```

If you are looking for general information about xPC Target, the xPC Target Embedded Option, a data sheet, or a list of supported I/O boards, then try the xPC Target Product page:

```
http://www.mathworks.com/products/xpctarget/
```

Technical Support

The Technical Support knowledge base is a database on the MathWorks Web site. It provides the answers to questions by users that are not answered in the documentation.

To Access Technical Support from the Help Browser

1. In the MATLAB window, and from the **Help** menu, click **Full Product Family Help**.

   The Help browser window opens.

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

   In the right pane, the Help browser displays the xPC Target Roadmap page.

3. Under the section titled “Web Pages,” select “Technical Support.”

   The Help browser searches the Tech Support Knowledge Base and displays entries specific for xPC Target.

   Note You need to be connected to the Internet for this search to be successful.

To Access Technical Support from a Web Browser

1. Open a Web browser.

2. In the address box, enter

   ```
   http://www.mathworks.com/search/
   ```

3. Select the **Tech Support Knowledge Base** check box. In the **Search for** box, enter `xpc`, and then click **Search**.

Using This Guide

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

- **Expected Background** — Proficiency with using MATLAB and Simulink, and familiarity with Real-Time Workshop
- **Organization** — Table with a list of chapters in the xPC Target Getting Started Documentation

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 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 Tutorial.”

If you are an experienced user — After you are familiar with using xPC Target, read or browse the following chapters in the xPC User’s Guide documentation: Chapter 4, “Software Environment,” Chapter 5, “Target Objects,” and Chapter 6, “Scope Objects” for more detailed information about the commands in xPC Target.

Organization

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

Chapter	Description
Preface	List of required and related products, suggestions for installing and printing the documentation, and places to find additional information.
Chapter 1, “Introduction”	Overview of the functions and features of xPC Target.
Chapter 2, “Installation and Configuration”	Install xPC Target on the host computer, and configure the host computer for communication with the target computer.
Chapter 3, “Basic Tutorial”	Create a boot disk with the xPC Target kernel, create a target application, and then download it to the target computer.
Chapter 4, “Signals and Parameters”	Learn some basic methods for changing parameters and acquiring signal data from a target application.

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 a target application from your Simulink model, compiling, linking, and downloading the generated code to create a *target application*

Term	Definition
execution	Running the *target application* on the target PC in real time
executable code	See target application
kernel	Real-time software component running on the target PC that manages the downloaded *target application*
model	Simulink/Stateflow model
parameter tuning	Process of changing block parameters and downloading the new values to a *target application* while it is running or not running
sample rate	Rate the *target application* is stepped in samples/second. Reciprocal of the *sample time*
sample time	Interval, in seconds, between the execution of a *target application* 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 and Stateflow model on the host PC in nonreal time
target application	Executable code generated from a Simulink and Stateflow model, which can be executed by the xPC Target kernel on the target PC

Typographical Conventions

This manual uses some or all of these conventions.

Item	Convention	Example
Example code	`Monospace` font	To assign the value 5 to `A`, enter `A = 5`
Function names, syntax, filenames, directory/folder names, and user input	`Monospace` font	The `cos` function finds the cosine of each array element. Syntax line example is `MLGetVar ML_var_name`
Buttons and keys	**Boldface** with book title caps	Press the **Enter** key.
Literal strings (in syntax descriptions in reference chapters)	**`Monospace bold`** for literals	`f = freqspace(n,'whole')`
Mathematical expressions	*Italics* for variables Standard text font for functions, operators, and constants	This vector represents the polynomial $p = x^2 + 2x + 3$.

MATLAB output	`Monospace` font	MATLAB responds with
		`A =`
		`5`

Menu and dialog box titles	**Boldface** with book title caps	Choose the **File Options** menu.
New terms and for emphasis	*Italics*	An *array* is an ordered collection of information.
Omitted input arguments	(...) ellipsis denotes all of the input/output arguments from preceding syntaxes.	`[c,ia,ib] = union(...)`
String variables (from a finite list)	*`Monospace italics`*	`sysc = d2c(sysd,'method')`

1

Introduction

This chapter is an overview of the functions and features of xPC Target. 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:

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** — Signal monitoring, signal 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 boot disk that includes the highly optimized xPC Target kernel.

However, the xPC Target Embedded Option requires DOS and a DOS license at boot time. For more information, see the xPC Target User Guide documentation.

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 are finished with 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, you can suppress checking 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 Real-Time Workshop Parameters” on page 3-26.

Real-Time Application

Real-Time Workshop, Stateflow Coder, xPC Target, and a C compiler create a real-time application (target application) from a Simulink and Stateflow model. Target applications created with Real-Time Workshop and xPC Target run in real time on a standard PC without using a Windows 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 executable is then converted to an image suitable for xPC Target, and it 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 µs (100 kHz). Model size, complexity, and target PC hardware affect maximum speed (minimal sample time) of execution.

For more information on creating a target application, see “xPC Target Application” on page 3-24.

Signal Acquisition

The xPC Target real-time kernel stores signal data from the target application in RAM on the target PC. You can use this signal data to analyze and visualize signals. 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 of acquiring signal data while a target application is running, and then visualizing the collected data after the target application stops running. 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 signal data while a target application is running. 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, "Signal Monitoring with MATLAB" on page 4-2, "Signal Logging" on page 4-18, and "Signal Tracing" on page 4-3.

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 tuning of parameters while the target application is running in real time.

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

- **Scripts and batch procedures** — xPC Target also includes commands to change parameters during a run or between runs. By writing and running a script on the host PC 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 and “Parameter Tuning” on page 4-24.

Hardware Environment

The hardware environment consists of a host computer, target computer, I/O boards in the target computer, and a serial or network connection between the host and target computers. Knowing the different types of computers and I/O supported by xPC Target will help you to setup a development environment that meets your needs. 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
- **I/O Driver Support** — Analog, digital, CAN, GPIB, RS232, UDP, counters, timers, and signal conditioning

Host PC

You can use any PC that runs a Microsoft Windows platform supported by The MathWorks as 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 2-2.

Target PC

You can use virtually any PC with an Intel 386/486/Pentium or AMD K5/K6/Athlon processor as the target computer. 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 reboot your computer without the target boot disk, and resume normal use of your desktop computer.

- **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 flash memory.

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

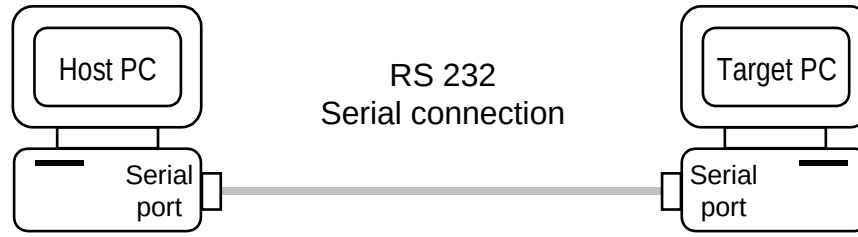
You do not need any special target hardware. However, the target PC must be a fully PC compatible system and contain a serial port or an Ethernet controller compatible with xPC Target.

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

Host-Target Connection

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

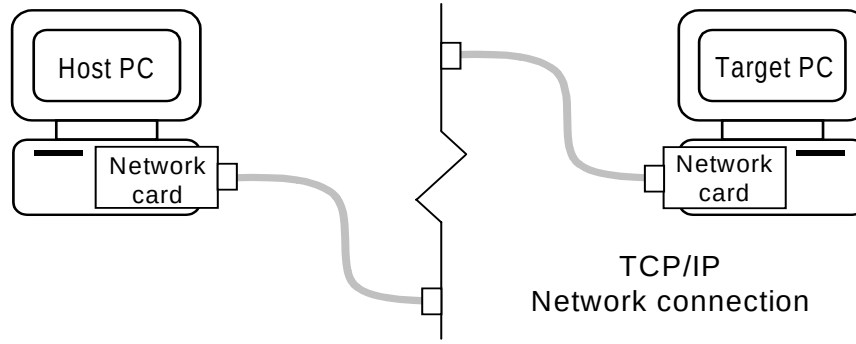
Serial — The host and target computers are connected directly together with a serial cable using their RS232 ports. This cable is wired as a **null modem** link that can be up to 5 meters long and with a transfer rate between 1200 and 115200 Baud. We provide a null modem cable with the xPC Target software.



For detailed information on setting up the hardware and software for serial communication, see "Serial Communication" on page 2-11.

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 using 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 can be 10 megabits/second or 100 megabits/second. For a list of supported cards, see "Hardware for Network Communication" on page 2-14.



For detailed information on setting up the hardware and software for network communication, see "Network Communication" on page 2-14.

I/O Driver Support

xPC Target supports a wide range of I/O boards. The list of supported I/O boards includes ISA, PCI, PC/104, 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 blocks from the I/O library and connect I/O drivers to your model the same way as you would connect any standard Simulink block.

I/O support — The I/O device library supports over 150 standard boards. I/O boards plug into the target PC expansion bus, PC104 stack, or industrial PC chassis. Also, there is support for modules that plug into IP or PMC carrier boards. 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** — Use the COM1 or COM2 ports for serial communication with external devices. See Chapter 1, “Serial Communications Support” in the xPC Target I/O Reference documentation.
- **CAN support** — You can use CAN-AC2, CAN-AC2-PCI, and CAN-AC2-104 boards from Softing GmbH AG with xPC Target CAN drivers to interface with a CAN fieldbus network. This interface provides communication through a CAN network between target 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. See Chapter 3, “CAN I/O Support” and Chapter 4, “CAN I/O Support for FIFO” in the xPC Target I/O Reference documentation.
- **GPIB support** — Special RS232 drivers support communication with a GPIB control module from National Instruments to external devices with a GPIB connector. See Chapter 2, “GPIB I/O Support” in the xPC Target I/O Reference documentation.

- **UDP support** — Communicate with another system using the standard UDP/IP network protocol. See Chapter 5, “UDP I/O Support” in the xPC Target I/O Reference documentation.
- **Counter-Timers** — Use the counter-timer blocks for measuring pulse and frequency with modulation applications.
- **Watchdog** — Monitor an interrupt or memory location, and reset the computer if an application does not respond. See “WDG-CSM” and “VSBC-6” in the xPC Target I/O Reference documentation.
- **Incremental encoder** — Change motion into numerical information for determining position, direction of rotation, and velocity.
- **Shared memory** — Use shared memory blocks with multiprocessing applications.

For information on using specific I/O driver blocks and advanced I/O support, see the xPC Target I/O Reference documentation.

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, and download the target application to a target PC. Run and test the target application with tuning of parameters and acquisition of data.
- **xPC Target Embedded Option** — 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 target 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 control of the target application, acquisition of signal data, and tuning of parameters while running in real time.

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

The rapid prototyping process includes the following sequence of tasks:

1. **Create a Simulink and Stateflow model** — You create block diagrams in Simulink using simple drag-and-drop operations, and 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 an xPC Target 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** — Acquire signal data using xPC Target Scopes. You can create xPC Target Scopes and acquire data from the target application by

- xPC Target — Using the xPC Target Remote Control Tool and Scope Manager window to create scopes, and using the Simulink viewer to add signals.
- MATLAB — Entering commands in the MATLAB window
- Simulink — Adding xPC Target Scope blocks to your Simulink model

  Note xPC Target does not support normal Simulink scope blocks in external mode. Instead, use xPC Target scope blocks
- Target PC — Using commands in the target PC command window
- Web browser — Using the xPC Target Web browser interface
- Simulink GUI — Adding blocks to a Simulink user interface model with xPC Target From blocks. See Chapter 2, “Graphical User Interfaces” in the xPC Target User’s Guide documentation for details.

6 **Tune parameters** — You can tune parameters by

- xPC Target — Using the xPC Target Remote Control Tool and Simulink viewer to change parameters
- MATLAB — Entering commands in the MATLAB window
- Simulink — Using your Simulink model with external mode
- Target PC — Using commands in the target PC command window
- Web browser — Using the xPC Target Web browser interface
- Simulink GUI — Adding blocks to a Simulink user interface model with xPC Target To blocks. See Chapter 2, “Graphical User Interfaces” in the xPC Target User’s Guide documentation for details.

xPC Target Embedded Option

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 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 on the target PC, or optionally boot from a device other than the floppy disk. The target application runs completely independent from the host PC.

When using the normal `BootFloopy` mode, you do not need DOS to load and run the xPC Target kernel.

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

For more information on the xPC Target Embedded Option, see the xPC Target documentation.

DOSLoader Mode

The DOSLoader mode allows you to boot your target computer from devices other than a 3.5-inch floppy disk. 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, but there is no advantage to using this method over using a normal target boot disk. The advantage to using the DOSLoader mode with the extra work to create a bootable DOS disk is that you can boot from an alternative device.

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, temporarily remove the line that executes `xpcboot.com`.
4. Boot the target PC with a bootable DOS disk from the floppy disk.
5. Copy files to the alternate boot device. In the `autoexec.bat` file, add 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 Chapter 3, “Embedded Option” in the xPC Target User Guide documentation.

(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, 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 a alternate boot device, in the `autoexec.bat` file, remove the line that loads `xpcboot.com`.

4 Boot the target PC with a bootable DOS disk from the floppy disk.

   Note If booting from an alternate boot device, copy files to the alternate boot device. In the `autoexec.bat` file, add 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 Chapter 3, “Embedded Option” in the xPC Target User Guide documentation.

User Interaction

The xPC Target environment has a modifiable interface to the target PC. You can use this interface from MATLAB or Simulink, and you can use other development environments to create stand-alone client applications independent of MATLAB. Because of this open environment, there are several ways to interact with your target application from the host and target PCs. This section includes the following topics:

- **xPC Target Remote Control Tool** — Use the xPC Target Remote Control Tool to set xPC Target environment properties, control the target application, add xPC Target Scopes, and change tunable parameters
- **MATLAB Command-Line Interface** — Enter xPC Target functions on the host PC
- **Simulink External Mode Interface** — Connect a Simulink block diagram to the target application using Simulink External Mode, and then use the block diagram for parameter tuning
- **Simulink with xPC Target Scope Blocks** — Add xPC Target Scope blocks to your model for signal tracing
- **Target PC Command-Line Interface** — Enter xPC Target functions on the target PC
- **Web Browser Interface** — Use Microsoft Internet Explorer or Netscape Navigator to connect to the target PC from any computer on the network with the target PC
- **Custom GUI with xPC Target API** — Create a client application that interfaces with a target application using any development environment that can call functions from a DLL
- **Custom GUI with xPC Target COM API** — Create a client application that interfaces with a target application using Visual Basic or any development environment which can incorporate COM objects

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

Interface	Environment properties	Control	Signal Acquisition	Parameter Tuning
xPC Target Remote Control Tool	X	X	X	X
MATLAB Command-Line Interface	X	X	X	X
Simulink External Mode Interface		X		X
Simulink with xPC Target Scope Blocks			X	
Target PC Command-Line Interface		X	X	X
Web Browser Interface		X	X	X
Custom GUI with xPC Target API		X	X	X
Custom GUI with xPC Target COM API		X	X	X

xPC Target Remote Control Tool

xPC Target offers a graphical user interface (GUI) for interacting with a target application. This GUI is built using xPC Target functions and MATLAB Handle Graphics®. To open the xPC Target GUI, in the MATLAB Command Window, type `xpcrctool`.

The xPC Target Remote Control Tool 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, type and location of your C compiler, and target PC properties.

  For more information on environment properties, see "Serial Communication" on page 2-11 and "Network Communication" on page 2-14 and Chapter 4, "Software Environment" in the xPC Target User Guide documentation.
- **Control** — Build, download, and run a target application. Change start and sample times without regenerating code, and get statistical performance information during or after the last run.

  For more information, see "Control with xPC Target Remote Control Tool" on page 3-38.
- **Signal acquisition** — Use the xPC Target Simulink Viewer to interactively add scopes of type host or target, and add or remove signals. Use the xPC Target Scope Manager to set the triggering mode of a scope. A scope created on the target PC acquires data from the target application and stores the data for display on the host PC or target PC:
  - For scopes of type host — The Host Scope Manager periodically uploads data packets and displays them in a MATLAB figure.
  - For scopes of type target — The scope displays signal traces on the target PC screen. Because xPC Target uses highly optimized graphic routines, signal tracing has the same fast display and update rates that you normally observe when using a digital oscilloscope.

  For more information on using scopes with the xPC Target Remote Control Tool, see "Signal Tracing with xPC Target Remote Control Tool" on page 4-3 and "Signal Logging with xPC Target Remote Control Tool" on page 4-18.
- **Parameter tuning** — Use the xPC Target Simulink Viewer to change tunable parameters in your target application.

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

For more information, see “Parameter Tuning with xPC Target Remote Control Tool” on page 4-24.

MATLAB Command-Line Interface

You can interact with the xPC Target environment through the MATLAB command-line interface. Enter xPC Target functions in the MATLAB Command Window on the host PC. Also, you can write your own M-file scripts that use xPC Target functions for batch processing.

xPC Target has more than 90 MATLAB functions for controlling the target application from the host computer. These functions define, at the most basic level, what you can do with the xPC Target environment.

The GUIs that we provide with xPC Target are for completing the most common tasks. They use the xPC Target 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** — Create a boot disk and directly change the environment properties without using a graphical interface.

  For more information on environment properties, see “Creating a Target Boot Disk with Command-Line Interface” on page 2-25, and Chapter 4, “Software Environment” in the xPC Target User Guide documentation
- **Control** — Reboot the target PC, download a target application, start and stop target applications, and change start and sample times without regenerating code. Get statistical performance information during or after the last run. Add/remove scopes, add/remove signals to scopes, and define triggers for scope display.

  For more information, see “Control with MATLAB Commands” on page 3-41 and Chapter 5, “Target Objects” in the xPC Target User Guide documentation.

- **Signal acquisition** — Trace signals for viewing while the target application is running and monitor signal values without time information. Transfer logged signal data to the MATLAB workspace by uploading from the host PC to the target PC between runs.

  For more information, see “Signal Monitoring with MATLAB” on page 4-2, “Signal Tracing with MATLAB” on page 4-13, and “Signal Logging with MATLAB” on page 4-20, and Chapter 6, “Scope Objects” in the xPC Target User Guide documentation.
- **Parameter tuning** — Change parameters while the target application is running, and use xPC Target functions to change parameters in between runs.

  For more information, see “Parameter Tuning with MATLAB” on page 4-26, and Chapter 5, “Target Objects” in the xPC Target User Guide documentation.

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 user 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** — Control is limited to connecting the Simulink block diagram to the target application, and starting and stopping the target application.

  For more information, see “Control with Simulink External Mode” on page 3-42.
- **Parameter tuning** — Select external mode, and change parameters in the target application by changing parameters in the **Block Parameters** dialog boxes. Once you change a value and press **OK**, the new value is downloaded to the target PC and replaces the existing parameter while the target application continues to run.

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

For more information, see “Parameter Tuning with Simulink External Mode” on page 4-28.

Note xPC Target does not support data acquisition through Simulink external mode. Instead, use xPC Target Scope blocks or signal acquisition. See “Simulink with xPC Target Scope Blocks” on page 1-23.

Simulink with xPC Target Scope Blocks

An alternative to interactively adding scopes to the target PC is to add xPC Target Scope blocks to your Simulink model. After the download process, these blocks create scopes on the target PC during initialization of the target application. You can choose to display data on either the host PC or target PC.

Signal acquisition — Add scopes to the target PC by adding xPC Target Scope blocks to your Simulink model. In the **Block Parameters** dialog box, select the scope mode and set the trigger.

For more information, see “Adding an xPC Target Scope Block” on page 3-11, “Entering Parameters for an xPC Target Scope Block” on page 3-15, and “Signal Tracing with xPC Target Scope Blocks” on page 4-15.

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, and change the stop time and sample time.

  For more information, see “Target PC Command-Line Interface” in the xPC Target User Guide documentation.
- **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.

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 the network. Also, if the target PC is connected to the host PC with an RS-232 cable, and is using the TCP/IP to RS-232 gateway, you can use a Web browser on the host PC.

The Web browser interface includes the following functions:

- **Control** — Start and stop the target application, and change the stop time and sample time.

  For more information, see Chapter 1, “Web Browser Interface” in the xPC Target User Guide documentation.
- **Signal acquisition** — Signal tracing is limited to viewing a snapshot of a 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.

  For more information, see “Signal Logging with a Web Browser” on page 4-23.
- **Parameter tuning** — Change parameters in an HTML form, and then submit that form to make the changes in your target application.

  For more information, see “Parameter Tuning with a Web Browser” on page 4-30.

Custom GUI with xPC Target API

Create a GUI application interface to a target application using any development environment which can link in a DLL.

Use the GUI application to control, tune parameters, and acquire signal data from a target application. The Custom GUI runs on the host PC and communicates with the target application on the target PC using RS-232 or TCP/IP communication. A GUI application can be a console or Windows application using ActiveX components.

For more information see the xPC Target API documentation.

Custom GUI with xPC Target COM API

Create a GUI application that interfaces with a target application using Visual Basic or any development environment that can incorporate COM objects.These COM objects connect graphic elements to parameters for parameter tuning, and they connect signals for acquiring data from your target application. To create a custom GUI application connected to an xPC Target application use the following process:

1. Optionally, tag parameters and signals in your Simulink model.
2. Build the target application, model specific COM library, and tagged parameter and signal COM libraries.
3. Create a GUI application that references the COM libraries.

For more information see the xPC Target API documentation.

2

Installation and Configuration

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 (p. 2-2)	Select a host PC and target PC
Installation on the Host PC (p. 2-6)	Get a valid license for xPC Target, and a separate license for the xPC Target Embedded Option. Install the xPC Target software from a CD or download from the Web
Serial Communication (p. 2-11)	Select RS232 communication for an easy and inexpensive installation
Network Communication (p. 2-14)	Select TCP/IP communication for faster data transfer rates and longer connections
Target Boot Disk (p. 2-23)	Boot the xPC Target kernel on the target PC and establish a connection with the host PC
Testing and Troubleshooting the Installation (p. 2-28)	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, PC/104+, or Compact PCI

Host PC

The host PC is usually your desktop computer where you install MATLAB, Simulink, Stateflow, Stateflow Coder, Real-Time Workshop, xPC Target, and xPC Target Embedded Option. A notebook computer is also a viable host PC because of its small size.

Software Requirements for the Host PC

The following table lists the minimum software xPC Target requires on your host PC. For a list of optional software products related to xPC Target, see the "Related Products" section in the Preface.

Software	Description
Operating system	A Microsoft Windows platform supported by The MathWorks
MATLAB	Version 6.5.2
Simulink	Version 5.2
Real-Time Workshop	Version 5.2
C language compiler	Microsoft Visual C/C++ versions 5.0, 6.0, or 7.0 Watcom C/C++ versions 10.6 or 11.0
xPC Target	Version 2.0.3

Hardware Requirements for the Host PC

The following table lists the minimum resources xPC Target requires on 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 later
Peripherals	Hard disk drive with 60 Mbytes of free space One 3.5-inch floppy disk drive CD-ROM drive
RAM	128 Mbytes or more

Target PC

The target PC has to be a PC compatible system. 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.

Software Requirements for the Target PC

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

Software	Description
Operating system	None. If you have an operating system installed on the target PC, the xPC Target kernel does not affect it.
BIOS	PC compatible

Hardware Requirements for the Target PC

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

Hardware	Description
Chip set	PC compatible with UART, programmable interrupt controller, keyboard controller, and counter
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. The xPC Target software includes a serial null modem cable and an Ethernet card for the target PC
CPU	Intel 386/486/ Pentium or AMD K5/K6/Athlon with or without a floating point processor or unit. We recommend a Pentium, Athlon or later CPU
Keyboard and mouse	Needed to control the target PC when you create stand-alone applications **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.
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.
Peripheral	One 3.5 inch floppy disk drive. A hard disk drive is not required. **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.
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 “I/O Driver Support” on page 1-10.

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 or as a file you download from the Web. This section includes the following topics:

- **Getting or Updating Your License** — Valid License File or Personal License Password (PLP)
- **CD Installation** — xPC Target version 2.0.3 on the MathWorks Release 13 SP2 CD
- **Web Download Installation** — xPC Target version 2.0.3 as a single file with the MathWorks installer from the Web
- **Setting Your Initial Working Directory** — MATLAB working directory outside of the MATLAB root directory

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 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

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, you can boot the target PC from a device other than a floppy disk and create stand-alone target applications separate from the host PC.

In addition, this option includes an API interface for creating your own custom GUI applications that connect to a target application. See the "xPC Target Embedded Option" on page 1-15.

CD Installation

We distribute xPC Target version 2.0.3 on the MathWorks Release 13 SP2 CD. For detailed information about the installation process, see the MATLAB Installation Guide for the Windows platform:

1. Insert the Release 13 CD into the host CD-ROM 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 drive.

2. Follow the instructions on each dialog box. From the product list, select the xPC Target check 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 2-11 or "Network Communication" on page 2-14.

Web Download Installation

We distribute xPC Target version 2.0.3 as a single file with the MathWorks installer from the Web. The remaining required software is distributed on the Mathworks Release 13 CD with the general installation program.

After you get a valid Personal License Password (PLP), you can install the xPC Target software. See “Getting or Updating Your License” on page 2-6:

1. In a 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. Select the Release 13 option, the **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, download all of the required files to a temporary directory.

   Your browser downloads the files `Installer.exe` and `xpc.cab` to your computer.

6. Double-click the file `Installer.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, and from the product list, select the xPC Target check 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 2-11 or “Network Communication” on page 2-14.

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.dlm`)

  Select a working directory outside of the MATLAB root. See "Setting Your Initial Working Directory" on page 2-9.
- **Real-Time Workshop Build directory** — The Real-Time Workshop 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:\MATLABROOT\Toolbox\rtw\targets\xpc:`

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

Setting Your 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 and Real-Time Workshop are mixed with the MATLAB directories. This mixing of files could cause you file management problems.

Setting 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, in the MATLAB Command Window, type

```
pwd
```

Setting from Within MATLAB

To temporarily set your MATLAB working directory, use the following procedure:

1 In the MATLAB Command Window, type

```
cd c:\<MATLAB working directory>
```

2 To check your working directory, type

```
pwd or cd
```

To permanently set working directory, see “Setting from the Desktop Icon” on page 2-9.

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** — Use a null modem cable to connect the host PC to the target PC
- **Environment Properties for Serial Communication** — Enter information about the C/C++ compiler you installed on the host PC and the COM port you connected to the null modem cable

For using network communication, see “Network Communication” on page 2-14.

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** — 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 port.
- **I/O boards** — If you use I/O boards on the target PC, you need to correctly install the boards. See the manufacturer’s literature for installation instructions.

Environment Properties for Serial Communication

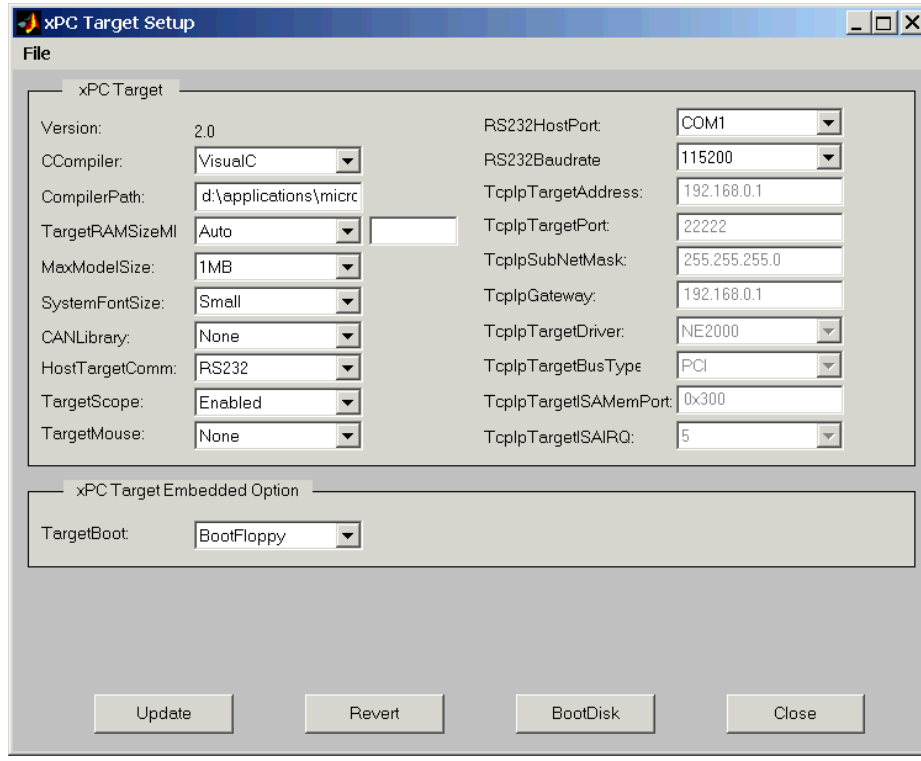
The xPC Target environment is defined by a group of properties. These properties give xPC Target information about the software and hardware products that it works with. You might change some of these properties often, while others you would change only rarely.

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:

1 In the MATLAB Command 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 `BootFloppy` 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, select 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, select `RS232`.

5 From the **RS232HostPort** list, select either `COM1` or `COM2` for the connection on the host PC. 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 Chapter 4, “Software Environment” in the xPC Target User Guide documentation.

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:

- Advantages of Network Communication
- 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 2-11.

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 100 Mbit/second 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 the host PC and the

target PC. Connect the host and target computers with an unshielded twisted pair (UTP) cable to your local area network (LAN).

Also, you can directly connect your computers together. Use a cross-over UTP cable with RJ45 connectors.

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

Ethernet Card Provided with xPC Target

The MathWorks supplies a PCI-bus Ethernet card with the xPC Target software for you to use in a desktop target PC. The following table will help to identify which card was shipped with your software and the parameter you need to select in the **xPC Target Setup** dialog box.

Board	Identification	Setup Parameter
SMC 1208BT	Card includes both BNC and RJ45 connectors and an SMC label on the packaging.	NE2000
Intel Pro/100S	Card has only an RJ45 connector and an Intel label on the back of the card.	I82559

1. In the MATLAB Command Window, type

   ```
   xpcsetup
   ```

   The **xPC Target Setup** dialog box opens.

2. From the **HostTargetComm** list, select `TCP/IP`.

3. From the **TcpIpTargetDriver** list, and depending on which Ethernet card was shipped with your software, select either `NE2000` or `I82559`.

4. Click the **Update** button.

Ethernet Cards Supported by xPC Target

xPC Target supports 100 Mbits/second Ethernet cards with

- **I82559** — Ethernet control chip from Intel for PCI bus cards with 10/100 Mbps. In the **xPC Target Setup** dialog box, and from the **TcpIpTargetDriver** list, select `I82559`.
- **AMD79C971 PCNET**— Fast Ethernet controller chips and other chips in the AMD 79C97x family. In the **xPC Target Setup** dialog box, and from the **TcpIpTargetDriver** list, select `RTLANCE`.

The Ethernet card included with xPC Target version 2.0.3 supports a data transfer rate of 100 Mbits/second, but we also support 10 Mbits/second Ethernet cards with

- **NE2000** — In the **xPC Target Setup** dialog box, select `NE2000`.
- **SMC91C9X** — In the **xPC Target Setup** dialog box, select `SMC91C9X`.

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

Board Type	Board Number	xPC Target Driver
	Intel PRO/100 S	I82559
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	RealTime 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 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 2-19.

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 2-14.

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, select 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 set up 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 2-19.

Environment Properties for Network Communication

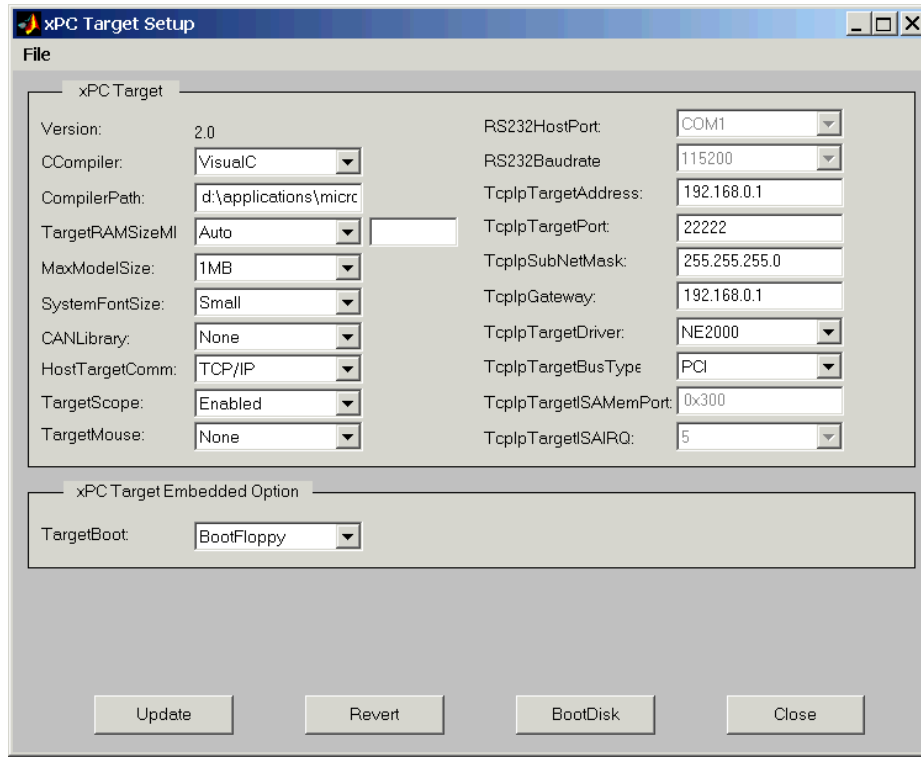
The xPC Target environment is defined by a group of properties. These properties give xPC Target information about the software and hardware that it works with. You change some of these properties often while others you change only rarely.

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:

1 In the MATLAB Command 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, select 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, select `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` and less then `65536`.
- **TcpIpGateway** — This property is set by default to `255.255.255.255`. This means that you 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** — From the list, select `NE2000`, `SMC91C9X`, `I82559`, `RTLANCE`. This property is set by default to `NE2000`.
- **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 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 Chapter 4, “Software Environment” in the xPC Target User documentation.

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

Target Boot Disk

The target boot disk includes the xPC Target kernel specific for either serial or network communication. Also, if you installed the xPC Target Embedded Option, and you select stand-alone mode, the target boot disk includes the target application. This section includes the following topics:

- Creating a Target Boot Disk with a Graphical User Interface
- Creating a Target Boot Disk with Command-Line Interface
- Current Properties on the Target Boot Disk

Creating a Target Boot Disk with a Graphical User 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. Alternatively, see “Creating a Target Boot Disk with Command-Line Interface” on page 2-25:

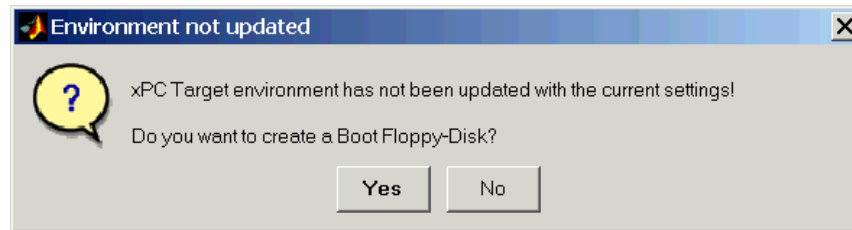
1 In the MATLAB Command Window, type

```
xpcsetup
```

The xPC Target Setup window opens.

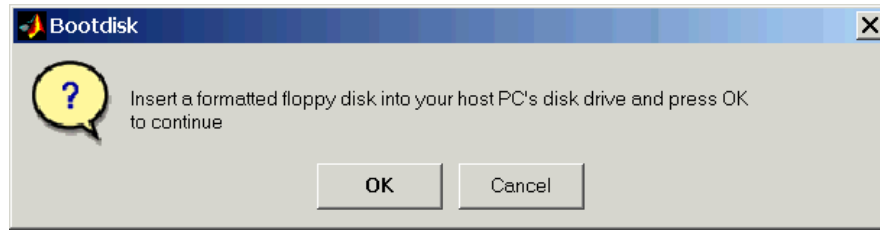
2 Click the **BootDisk** button.

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.

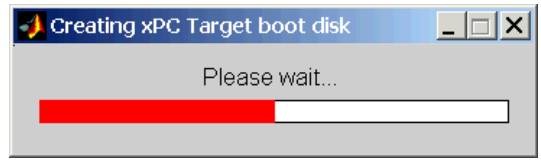
After you update the current properties, and click the **BootDisk** button, the following message box opens.



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 2-28.

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 Command 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.

```
Creating xPC Target boot disk ... Please wait

xPC Target boot disk successfully created.
```

Your next task is to test your installation. See “Testing and Troubleshooting the Installation” on page 2-28.

Current Properties on the Target Boot Disk

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

Note If you upgrade your xPC Target software, you need to recreate the target boot disk.

To Check the Target Boot Disk with an xPC Target GUI

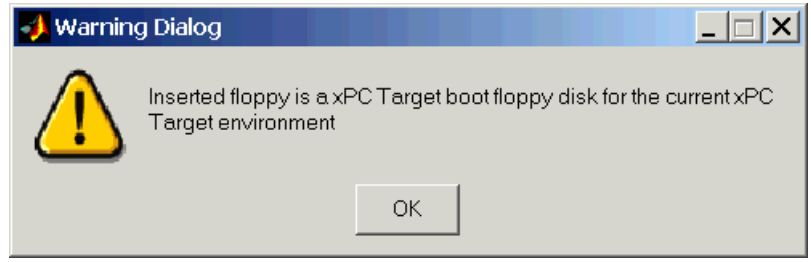
1 Insert the target boot disk into your host PC drive. In the MATLAB Command Window, type

```
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 warning appears 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. In the MATLAB Command Window, 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 no 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

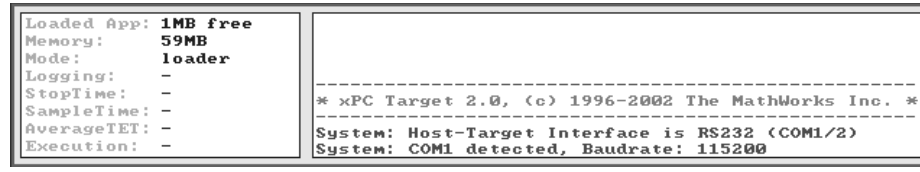
Testing the Installation

xPC Target uses a test script 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.



3 In the MATLAB Current Directory window, select a directory outside of the MATLAB root directory.

Note During the build process, Real-Time Workshop does not allow files to be saved within the MATLAB tree root. If you select a current directory within the MATLAB tree, the xPC Target test procedure will fail when trying to build a target application.

4 In the MATLAB Command 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 2.0.1
### 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 all of the tests were successful, you are ready to build and download a target application to the xPC TargetBox. See Chapter 3, “Basic Tutorial.”

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

- “Test 1, Ping Target System Standard Ping” on page 2-29
- “Test 2, Ping Target System xPC Target Ping” on page 2-31
- “Test 3, Reboot Target Using Direct Call” on page 2-32
- “Test 4, Build and Download Application” on page 2-32

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:
Replay form xxx.xxx.xxx.xxx: bytes-32 time<10 ms 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. On the host PC, open the **xPC Target Setup** dialog box, change the TCP/IP options, click the **Update** button, and then create a new boot floppy disk. On the target PC, reboot with the corrected boot floppy disk.

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 xpctest 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 2-33.

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 Command 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 2-33.

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 Command 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 could be that 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” on page 2-33.

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” on page 2-33.

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

3

Basic Tutorial

This chapter explains the basic functions of xPC Target by using a simple Simulink model. Since this model does not have I/O blocks, you can try these procedures regardless of whether you have I/O hardware on your target PC. This chapter includes the following sections:

Simulink Model

Before you can create a target application, you need to create a Simulink model. xPC Target then uses the Simulink model, Real-Time Workshop, and a third-party compiler to create the target application.This section includes the following topics:

- **Creating a Simple Simulink Model** — Create a simple second order system with a Sin wave input
- **Adding a Simulink Outport Block** — Add an Outport block to log signal data to the MATLAB workspace for analysis
- **Entering Parameters for Outport Blocks** — Enter and select parameters in the **Simulation Parameters** dialog box
- **Adding an xPC Target Scope Block** — Add an xPC Target Scope block to visualize signals while running the target application
- **Entering Parameters for an xPC Target Scope Block** — Enter scope parameters in the **Block Parameters** dialog box before building the target application

Creating a Simple Simulink Model

This tutorial uses a simple Simulink model to explain the tasks you need to do with xPC Target. If you are an experienced Simulink user, you can skip creating this model.

The model includes a transfer function and a signal generator block. If you want to visualize signals while simulating your model, you need to add a standard Simulink Scope block:

1 In the MATLAB Command Window, type

```
simulink
```

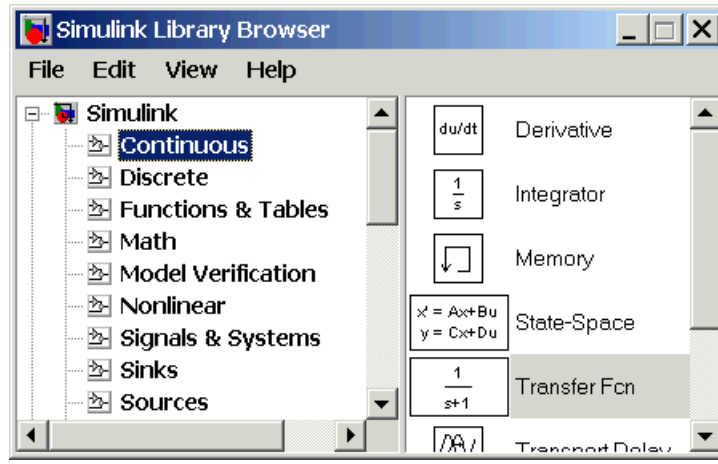
The Simulink library window opens.

2 From the **File** menu, point to **New**, and then click **Model**.

A blank Simulink model window opens.

3 In the left pane, double-click **Simulink**, and then click **Continuous**.

In the right pane, the Simulink library shows a list of blocks.



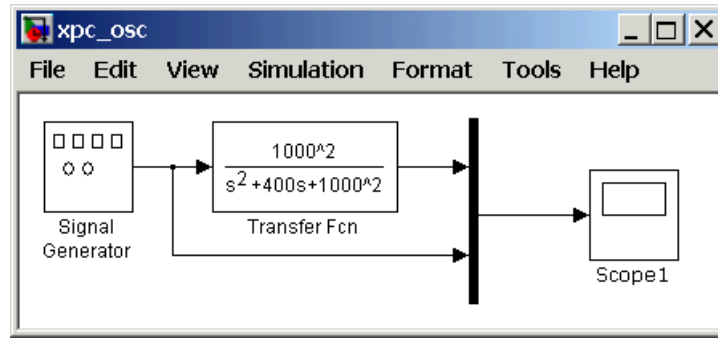
4 Click-and-drag the Transfer Fcn block to the Simulink model window.

5 In the Simulink Library Browser window, click-and-drag the following blocks to your model.

- Click Sources, and add a Signal Generator block
- Click Sinks, and add a Scope block
- Click Signals & Systems, and add a Mux block

6 Connect the Signal Generator block to the Transfer Fun block, and connect the input and output signals to the scope block using the Mux block.

Your model should look similar to the figure shown below.



7 From the **File** menu, click **Save as** and enter a filename. For example, enter `xpc_osc` and then click **OK**.

Since xPC Target does not support data uploading with external mode, you cannot use the Simulink Scope block to visualize signals from an xPC Target application running in real time. Instead, use an xPC Target Scope block. See "Adding an xPC Target Scope Block" on page 3-11.

For information on creating a Simulink model and adding signal and scope blocks, see the Using Simulink documentation.

Adding a Simulink Outport Block

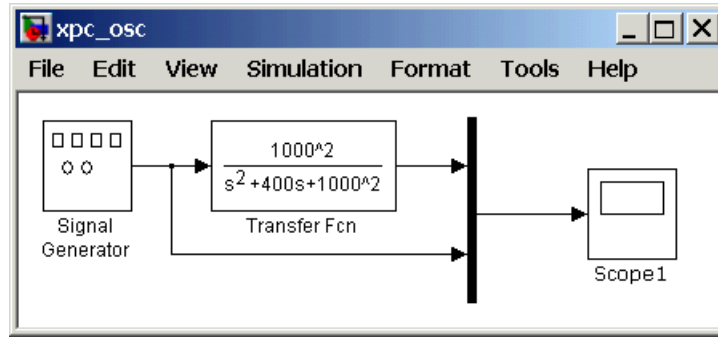
If you want to log signal data to the MATLAB workspace for analysis and later save that data to a disk, you need to add a Simulink Outport block and activate logging from the **Simulation Parameters** dialog box.

The following procedure uses the Simulink model `xpc_osc.mdl` as an example. To create this model, see "Creating a Simple Simulink Model" on page 3-2.

1 In the MATLAB Command Window, type

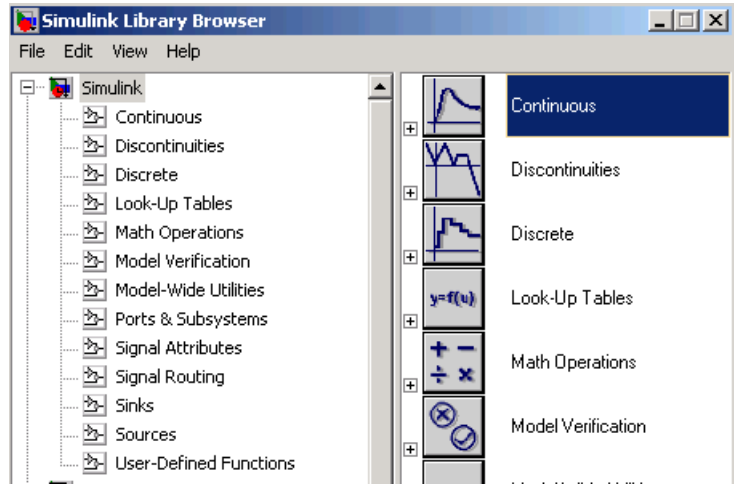
```
xpc_osc
```

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



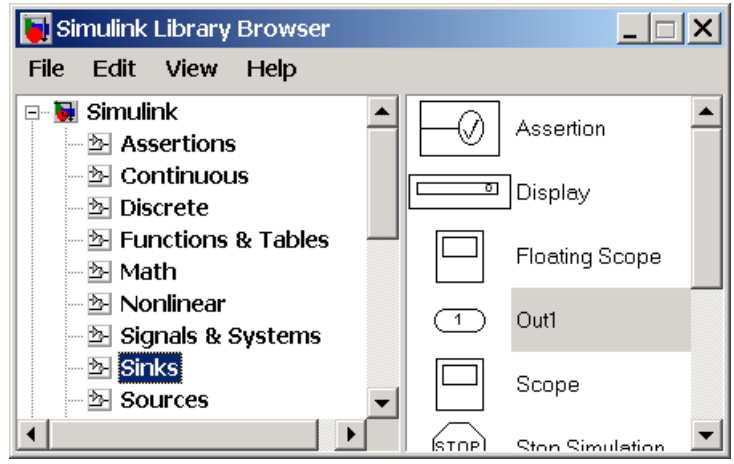
2 In the Simulink window, from the **View** menu, click **Library Browser**.

The **Simulink Library Browser** window opens.



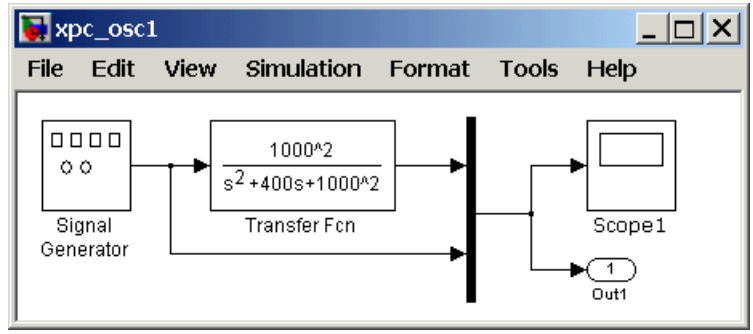
3 In the left pane, double-click **Simulink**, and then click **Sinks**.

In the right pane, the browser shows a list of sink blocks.



4 Click-and-drag the Out1 block to your model and connect it to the output of the Mux block.

Your model should look similar to the figure shown below.



5 From the **File** menu, click **Save as** and enter a filename. For example, enter `xpc_osc1` and then click **OK**.

Your next task is to enter parameters for the Outport block. See “Entering Parameters for Outport Blocks” on page 3-7.

Entering Parameters for Outport Blocks

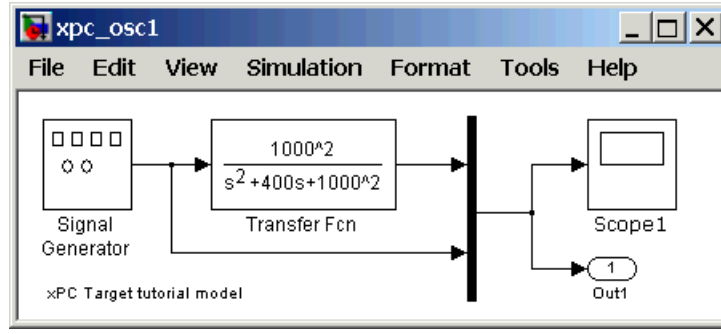
During a simulation, Simulink saves signal data to MATLAB variables using Outport blocks. The default MATLAB variables are `tout`, `xout`, and `yout`. While running a real-time application, xPC Target uses these same variables to pass signal data to target object parameters. A target object is a structure in the MATLAB workspace that xPC Target uses to interact with a target application. The default target object is `tg`, and the default parameters are `tg.Time`, `tg.States`, and `tg.Output`.

After you add an Outport block to your Simulink model, you can enter parameters. This procedure uses the model `xpc_osc1.mdl` with an outport block as an example. To add an Outport block, see "Adding a Simulink Outport Block" on page 3-4.

1 In the MATLAB Command Window, enter

```
xpc_osc1
```

A Simulink window with the model `xpc_osc1` opens.



2 In the Simulink window, from the **Simulation** menu, click **Simulation Parameters**.

The **Simulation Parameters** dialog box opens.

3 In the **Simulation Parameters** dialog box, click the **Solver** tab.

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

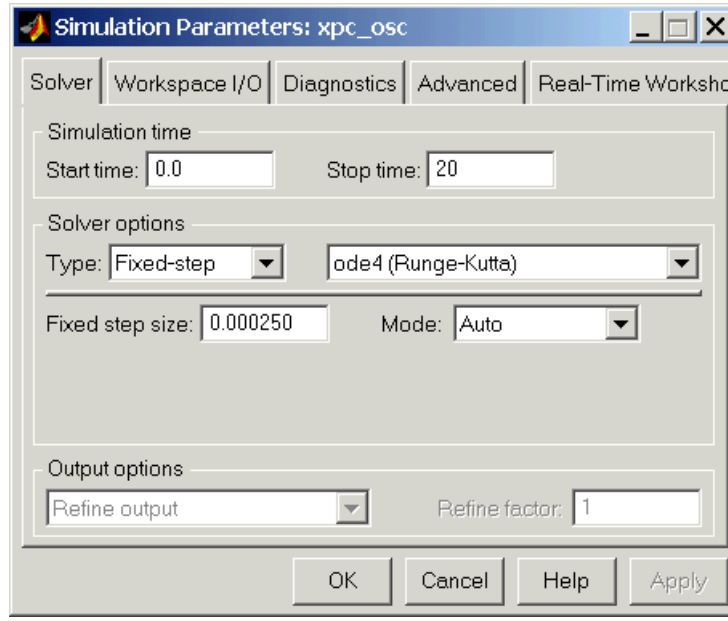
4 In the **Start time** box, enter `0` second. In the **Stop time** box, enter an initial stop time. For example, enter `20` seconds. To change this time after creating your target application, change the target object property `tg.Stoptime` to the new time using the MATLAB command-line interface.

5 From the **Type** list, select `Fixed-step`. Real-Time Workshop does not support variable step solvers. From the algorithm list, select a solver. For example, select 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` second (`250` microseconds). You can change this value after creating the target application.

   If you find that `0.000250` second 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 with both continuous and discrete states, 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 pane should look similar to the figure shown below.



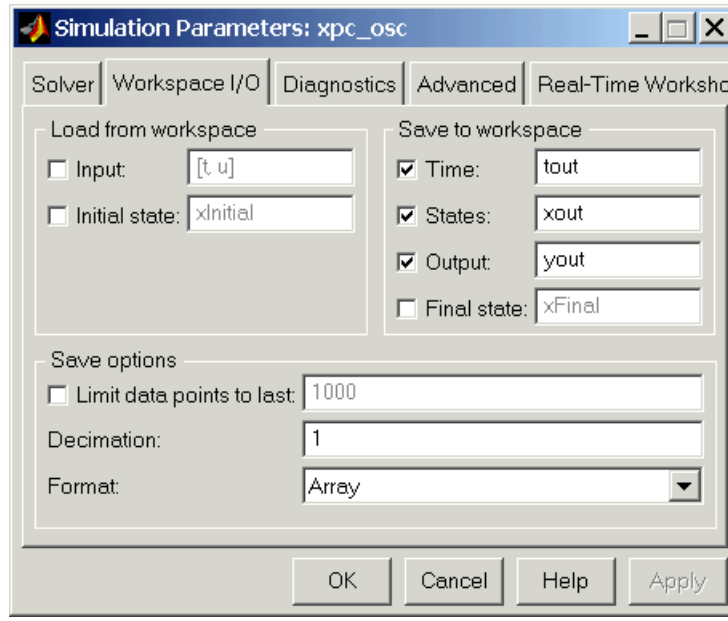
6 On the **Simulation Parameters** dialog box, click the **Workspace I/O** tab.

The Workspace I/O pane opens. This page defines which model signals are logged during a simulation of your model or while running your target application.

7 In the **Save to workspace** section, select the **Time**, **States**, and **Output** check boxes.

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 pane should look similar to the figure shown below.



Normally you select all of the **Save to workspace** check boxes. However, you may want to consider clearing some or all of them off 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 clearing the **States** check box, logging of states is turned off and 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 outport blocks to your model.
- **Small sample time** — If you choose a very short sample time that overloads the CPU. By clearing **Save to workspace** check boxes, logging is turned off and more computing time is available for calculating the model.

8 Click **OK**.

9 From the **File** menu, click **Save**. The model is saved as `xpc_osc1.mdl`.

Your next task is to add an xPC Target Scope block to your Simulink model. See “Adding an xPC Target Scope Block” on page 3-11.

Adding an xPC Target Scope Block

xPC Target does not support the standard Simulink Scope blocks, but it does support xPC Target Scope blocks, which have unique capabilities when you use them with an xPC Target application. These xPC Target Scope blocks should not be confused with standard Simulink 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. Also, the signal information is saved with your model.

After you create a Simulink model, you can add an xPC Target Scope block. The following procedure uses the Simulink model `xpc_osc1.mdl` as an example to show how to connect an xPC Target Scope block to your model:

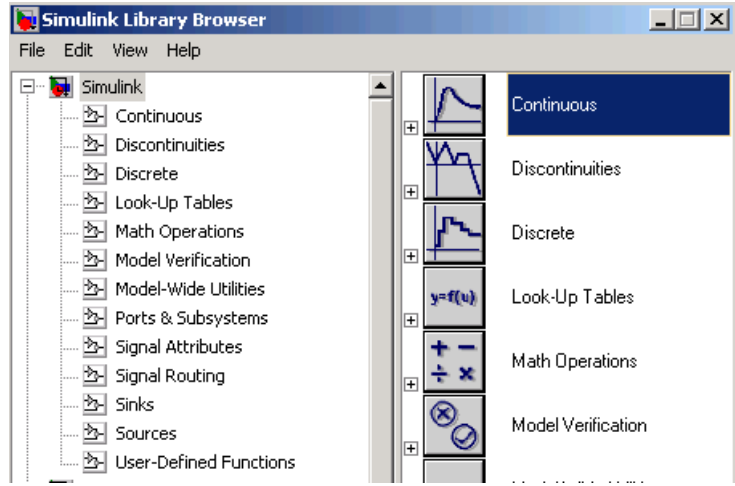
1. In the MATLAB Command Window, type

   ```
   xpc_osc1
   ```

   The Simulink block diagram opens for the model `xpc_osc1.mdl`.

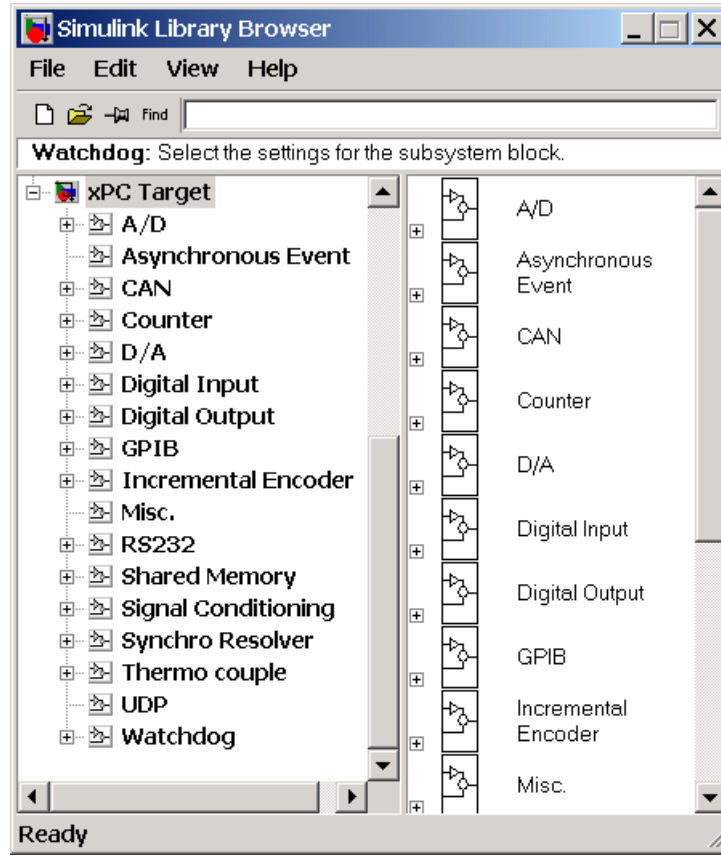
2. In the Simulink window, from the **View** menu, click **Show Library Browser**.

The **Simulink Library Browser** window opens.



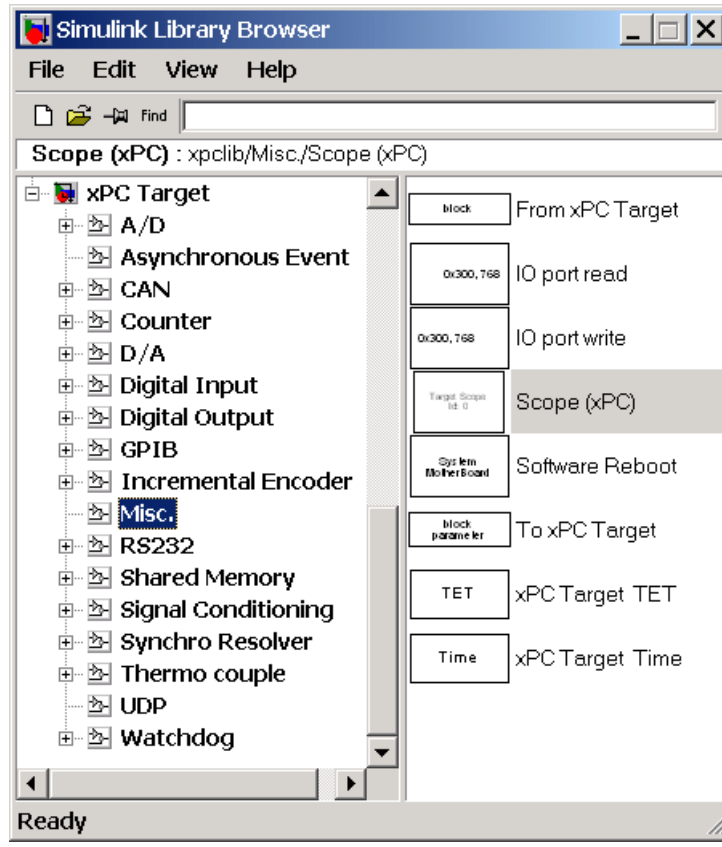
3 Double-click **xPC Target**.

A list of I/O functions opens.



4 Double-click **Misc**.

A list of miscellaneous group blocks opens.

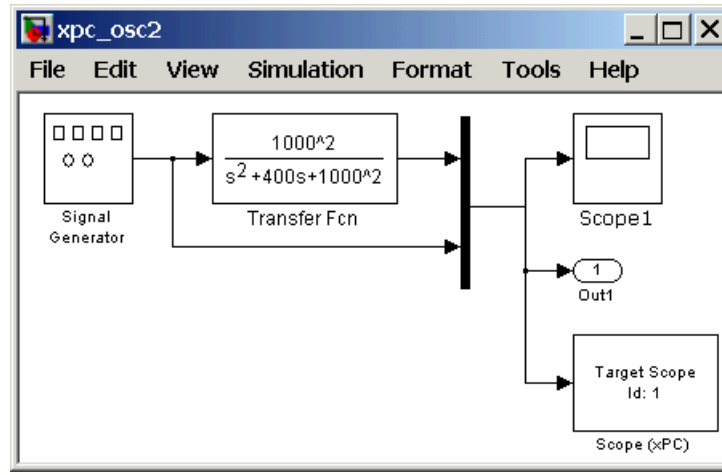


5 Click-and-drag **Scope (xPC)** to your Simulink block diagram.

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 `xpc_osc1.mdl` should look like the figure shown below.



7 From the **File** menu, click **Save as**. Enter a filename. For example, enter `xpc_osc2` and then click **OK**.

Your next task is to define the xPC Target Scope block parameters. See "Entering Parameters for an xPC Target Scope Block" on page 3-15.

Entering Parameters for an xPC Target Scope Block

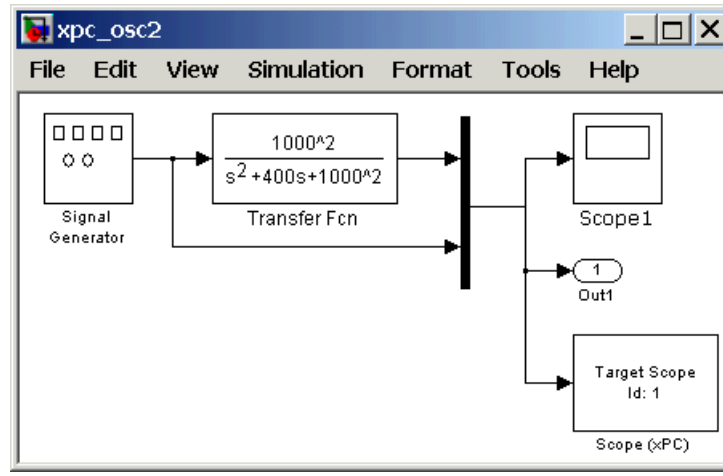
xPC Target Scope block parameters define the signals to trace on the scope and trigger modes. When the target application is downloaded to the target PC, the xPC Target kernel automatically creates the scope on the target PC. No additional definitions are necessary.

After you add an xPC Target Scope block to your Simulink model, you can enter parameters for this block. To add an xPC Target Scope block, see "Adding an xPC Target Scope Block" on page 3-11. This procedure uses the model `xpc_osc2.mdl` as an example:

1 In the MATLAB Command Window type

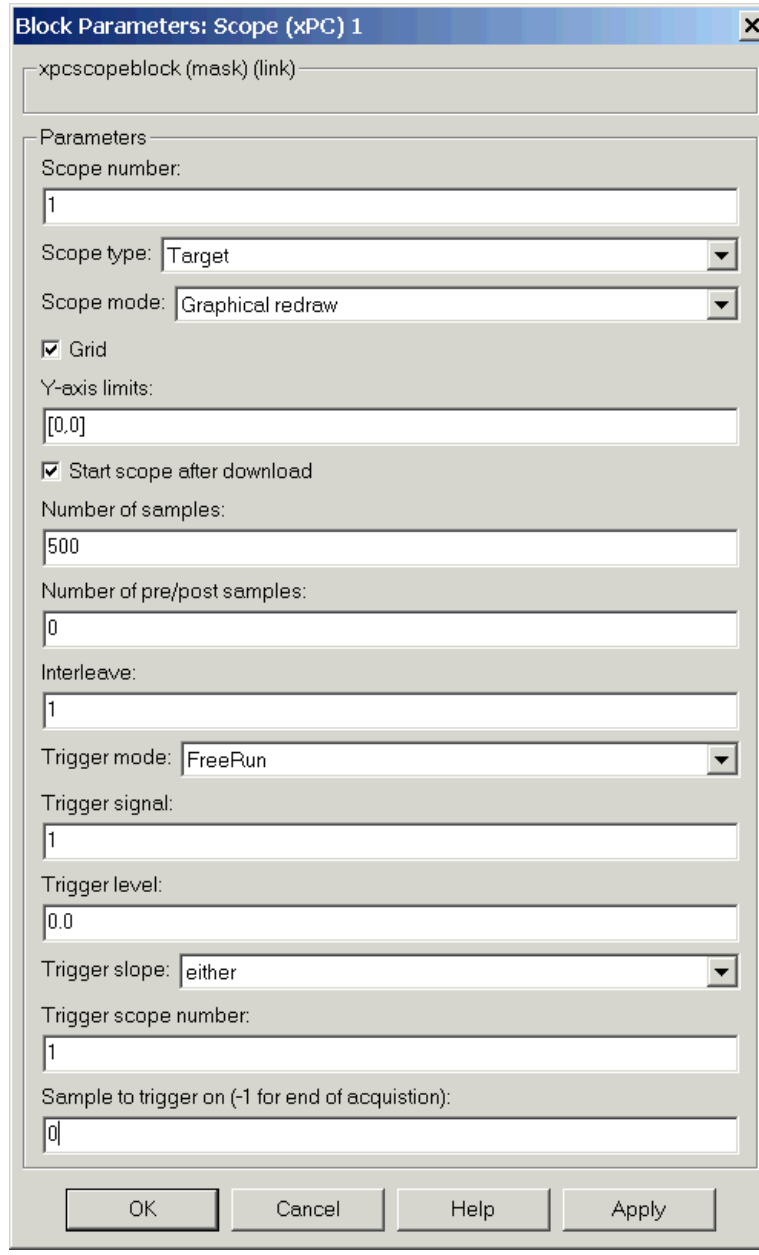
```
xpc_osc2
```

The Simulink block diagram opens for the model `xpc_osc2.mdl`.



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

The **Block Parameters: Scope(xPC)** dialog box opens.


Block Parameters: Scope (xPC) 1
xpcscopeblock (mask) (link)
Parameters
Scope number:
1
Scope type:
Target
Scope mode:
Graphical redraw
Grid
Y-axis limits:
[0,0]
Start scope after download
Number of samples:
500
Number of pre/post samples:
0
Interleave:
1
Trigger mode:
FreeRun
Trigger signal:
1
Trigger level:
0.0
Trigger slope:
either
Trigger scope number:
1
Sample to trigger on (-1 for end of acquistion):
0
OK
Cancel
Help
Apply

3 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.

4 From the **Scope Type** list, select either `Host` or `Target`.

5 From the **Scope Mode** list, select either `Numerical`, `Graphical redraw`, `Graphical sliding`, or `Graphical rolling`.

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

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

7 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.

8 Select the **Start Scope after download** check box to start a scope when the target application is downloaded. With a scope of type `target`, the scope window opens automatically. With a scope of type `host`, you need to open the window by typing `xpcscope`.

9 In the **Number of Samples** box, enter the number of values to be 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).

10 From the **Trigger Mode** list, select either `FreeRun`, `Software Triggering`, `Signal Triggering`, or `Scope Triggering`.

   If you select `Signal Triggering`, then in the **Trigger Signal** box, enter the index of a signal. In the **Trigger Level** box, enter a value for the signal to cross before triggering. From the **Trigger Slope** list, select `either` `rising` or `falling`.

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

If you want the scope to trigger on a specific sample of the other scope, enter a value in the **Sample to trigger on** box. The default value is `0` and indicates that the triggering scope and the triggered (current) scope start simultaneously. For more information on this field, see Chapter 6, "Using the Property TriggerSample to Capture Data" in the xPC Target User Guide documentation.

11 Click **OK**.

12 From the **File** menu, click **Save**. The model is saved as `xpc_osc2.mdl`.

Your next task is to simulate the model. See "Simulating the Model" on page 3-20.

Note As soon as the target application is built and downloaded, the xPC Target kernel creates a scope. If you want to change xPC Target Scope parameters after building the target application or while it is running, you need to assign it to a MATLAB variable. To assign the scope object, use the target object method `getscope`.

Also, if you use the target object method `remscope` to remove a scope created during the build and download process, and then you restart the target application, the xPC Target kernel recreates the scope.

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:

- **Simulating the Model with Simulink**— Use the Simulink window interface to run a simulation
- **Simulating the Model from MATLAB** — Use Simulink functions in the MATLAB Command Window to run a simulation

For procedures to run your target application in real time, see "Running the Target Application" on page 3-38.

Simulating the Model with Simulink

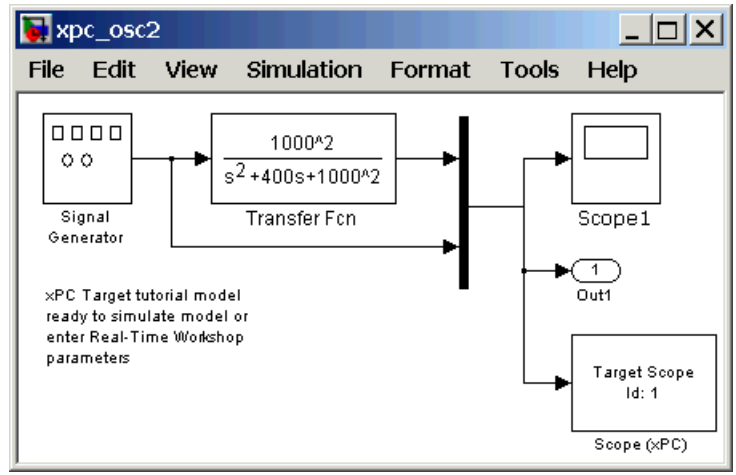
You run a simulation of your Simulink model to observe the behavior of the model in nonreal time.

After you load your Simulink model, you can run a simulation. This procedure uses the Simulink model `xpc_osc2.mdl` as an example and assumes you have already loaded that model. To create this model, see "Creating a Simple Simulink Model" on page 3-2.

1 In the MATLAB Command Window, type

```
xpc_osc2
```

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

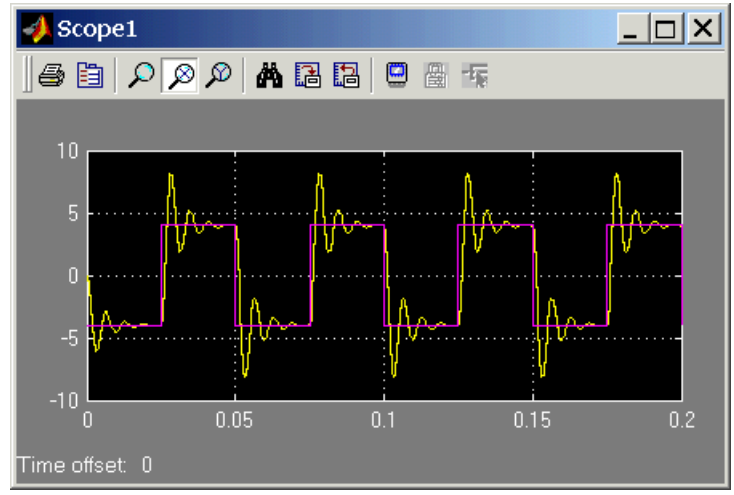


2 In the Simulink window, double-click the block Scope1.

Simulink opens a scope window.

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

The **Scope1** window displays a trace of the signal data.



4 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**.

Your next task is to create an xPC Target application. See "xPC Target Application" on page 3-24.

Simulating the Model from MATLAB

You run a simulation of your Simulink 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 `xpc_osc2.mdl` as an example and assumes you have already loaded that model. To create this model, see "Creating a Simple Simulink Model" on page 3-2.

1 In the MATLAB Command Window, type

```
sim('xpc_osc')
```

Simulink runs a simulation in normal mode.

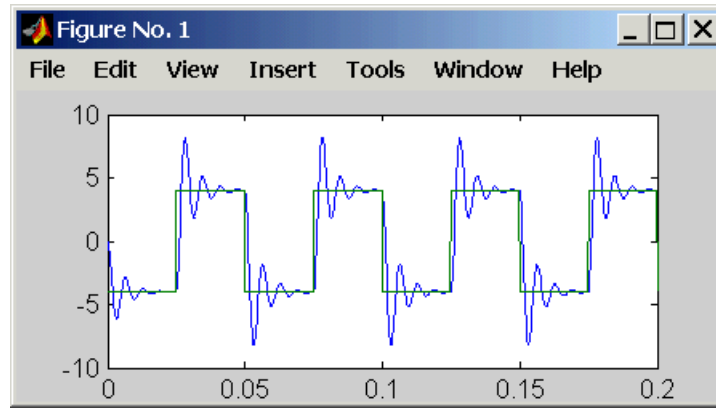
2 You can either let the simulation run to its stop time, or stop the simulation manually. To stop the simulation manually, press **Ctrl+T**.

3 After Simulink finishes the simulation, type

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

You entered the MATLAB variables `tout` and `yout` in the Workspace I/O pane on the **Simulation Parameters** dialog box. The signals are logged into memory through Outport blocks. To add an Outport block, see "Adding a Simulink Outport Block" on page 3-4 and "Entering Parameters for Outport Blocks" on page 3-7.

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 variables `tout` and `yout`. Instead, data is saved in the target PC memory and may be retrieved through the target object properties tg.`TimeLog`, tg.`StateLog`, and tg.`OutLog`. However, in the **Simulation Parameters** dialog box, 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 “xPC Target Application” on page 3-24.

xPC 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** — Start the xPC Target kernel running on the target PC
- **Troubleshooting the Boot Process** — Solutions to problems when booting the target PC
- **Entering the Real-Time Workshop Parameters** — Enter parameters specific for building target applications
- **Building and Downloading the Target Application** — Use Real-Time Workshop and a third-party C compiler to create the target application
- **Troubleshooting the Build Process** — Solutions to problems when building your target application
- **Increasing the Time Out Value** — Increase the time the download process waits for a target application to load onto the target PC
- **xPC Target Code Generation Options** — Reference for all of fields in the dialog box

For procedures to simulate your model in nonreal time, see "Simulating the Model" on page 3-20.

Booting the Target PC

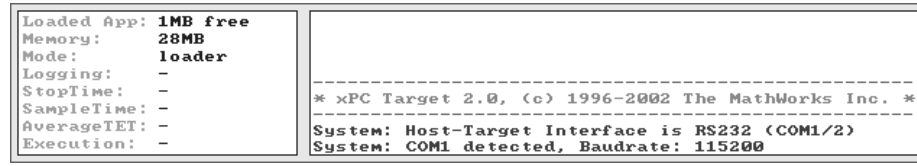
Booting the target computer loads and starts the xPC Target kernel on the target PC. The loader then waits for xPC Target to download your target application from the host PC.

After you have configured xPC Target using the **Setup** dialog box, 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.



In the example above, the status window shows the kernel is in the loader mode and waiting to load a target application. 1 MB of memory is reserved for the application, 3 MB is used by the kernel, and 28 MB is available from a total of 32 MB. The xPC Target kernel uses the 28 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 Real-Time Workshop Parameters” on page 3-26.

Troubleshooting the Boot Process

Possible problem — When booting the target PC, it might display the message

```
xPC Target 2.0.1 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 — Reformat the disk or use a new formatted floppy disk and create a new target boot disk.

Possible problem — When booting the target PC, you get a message similar to the following

```
Not a bootable medium or NTLDR is missing
```

This message can be caused by selecting either the `DOSLoader` or `StandAlone` mode instead of the `BootFloppy` mode.

Solution — Open the **xPC Target Setup** dialog box. In the MATLAB Command Window, type `xpcsetup`, and from the **TargetBoot** list, select `Bootfloppy`. Update the environment properties and then create a new boot floppy disk.

Entering the Real-Time Workshop Parameters

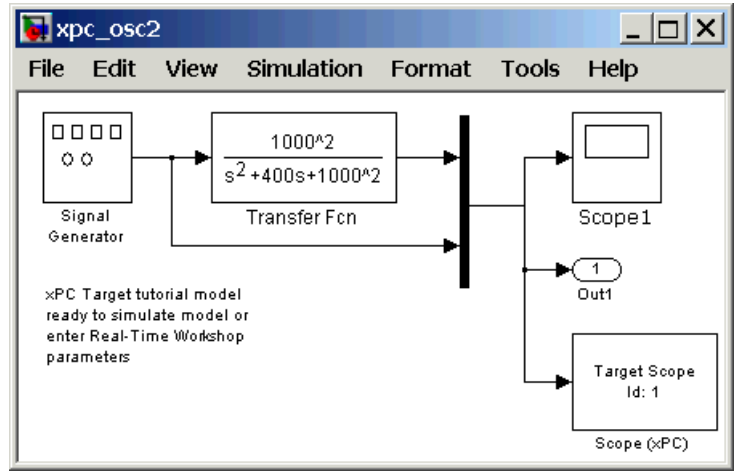
You enter the simulation and real-time run parameters in the **Simulation Parameters** dialog box. These parameters give information to Real-Time Workshop on how to build the target application from your Simulink model.

After you load a Simulink model and boot the target PC, you can enter the simulation parameters. This procedure uses the Simulink model `xpc_osc2.mdl` as an example and assumes you have already loaded that model. See "Simulink Model" on page 3-2.

1 In the MATLAB Command Window, type

```
xpc_osc2
```

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



2 In the Simulink window, and from the **Simulation** menu, click **Parameters**. In the **Simulation Parameters** dialog box, click the **Real-Time Workshop** tab.

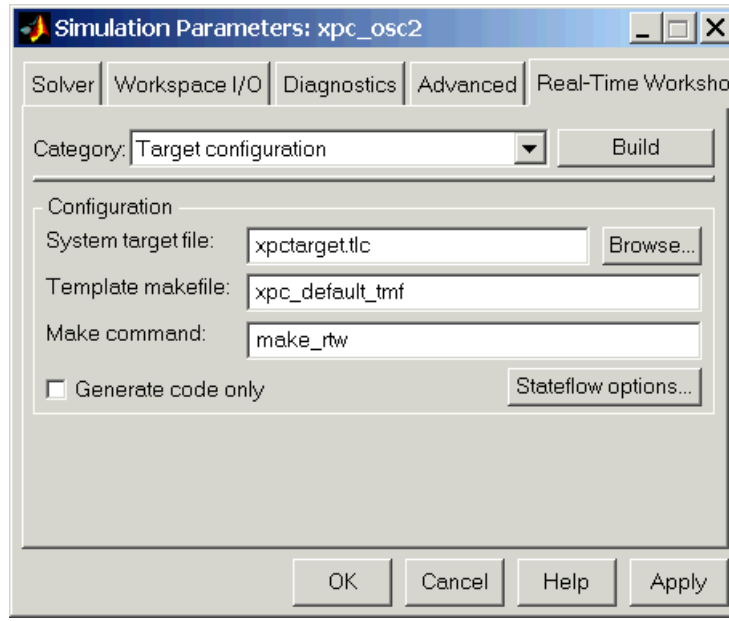
The Real-Time Workshop pane opens.

3 Click the **Browse** button.

The System Target File Browser opens.

4 From the list, select `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 pane should now look like the figure shown below.



5 From the **Category** list, select `xPC Target code generation options`.

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

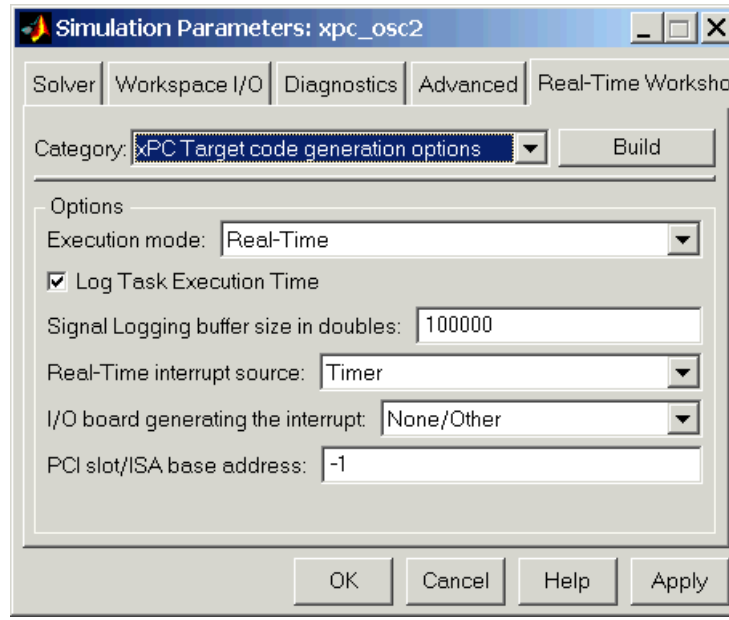
6 From the **Mode** list, select either `Real-Time` or `Freerun`. The option `Freerun` is similar to a simulation, but with the generated code. It runs the target application as fast as it can. However, unlike a simulation, the `Freerun` mode of xPC Target does not support variable step solvers.

7 Select the **Log Task Execution Time** check box to log task execution times to the target object property `tg.TETlog`.

The task execution time is the time in seconds to complete calculations for the model equations and post outputs during each sample interval.

8 In the **Signal Logging Buffer Size in Doubles** box, enter the maximum number of sample points to save before wrapping. This buffer includes the time, states, outputs, and task execution time logs.

9 From the **Real-Time Interrupt Source** list, select a source. The default value is `Timer`.



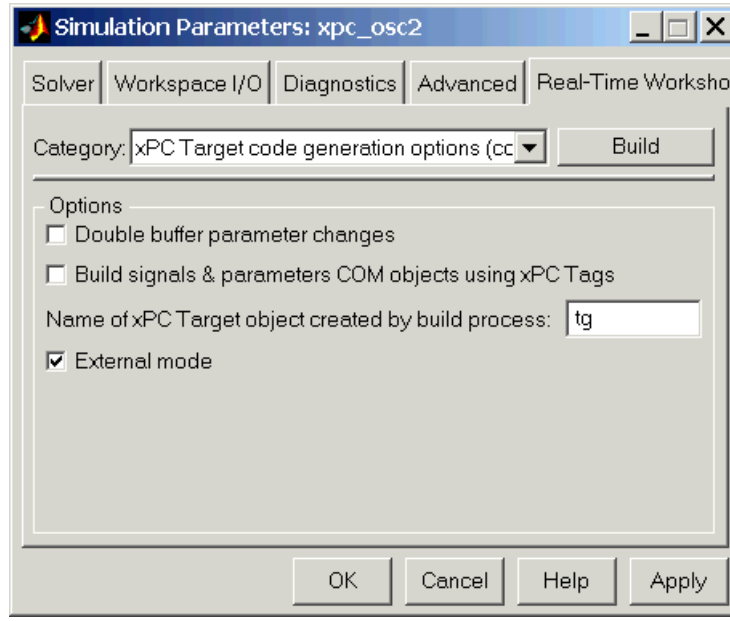
10 From the **Category** list, select `xPC Target code generation options (cont.)`.

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

11 In the **Name of xPC Target object** box, enter the name of the target object created by the build process. The default target object name is `tg`.

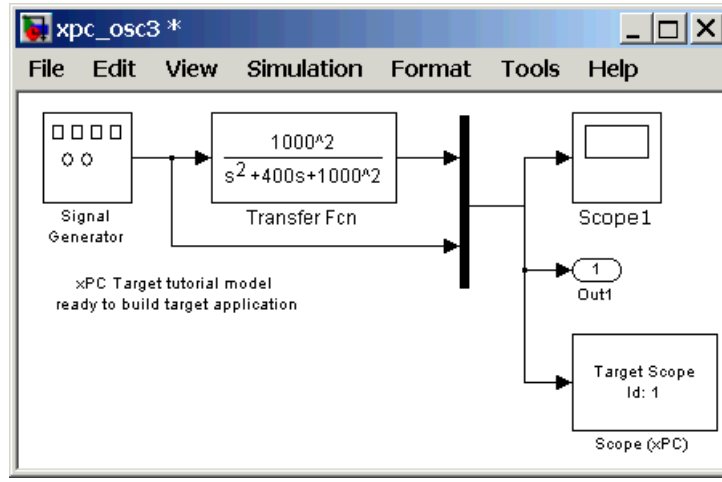
12 Select the **External mode** check box. The build process adds code to the target application to communicate with your Simulink model for parameter tuning.

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



13 Click **OK**.

14 From the **File** menu, click **Save as**. Enter a filename. For example, enter `xpc_osc3` and then click **OK**.



Your next task is to create (build) the target application. See “Building and Downloading the Target Application” on page 3-30.

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 `xpc_osc3.mdl` as an example. To create this model, see, “xPC Target Application” on page 3-24.

1 In the MATLAB Command Window, type

```
xpc_osc3
```

MATLAB loads the oscillator model and displays the Simulink block diagram.

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

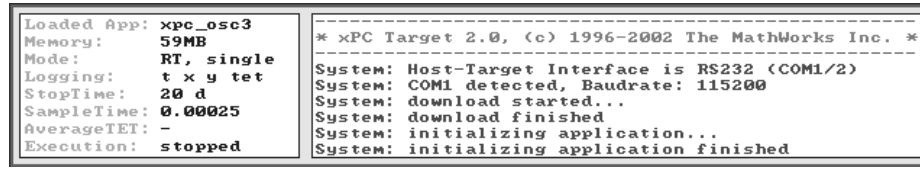
After the compiling, linking, and downloading process, a target object is created in your MATLAB workspace. The default name of the target object

is `tg`. For more information about the target object, see Chapter 5, "Target Objects" in the xPC Target User Guide documentation.

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.



3 In the MATLAB Command 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 3-32.

Your next task is to run the target application in real time on the target PC. See "Running the Target Application" on page 3-38.

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 Command Window, type

   ```
   xpcsetup
   ```

   The xPC Target **Setup** window opens.

2. Check RS232 or TCP/IP parameters. If necessary, make changes to the communication properties, update the properties, and recreate the target boot disk.

   In some cases, the download may have completed successfully even though you get a time-out error. To detect this, wait until the target screen displays

   ```
   System:initializing application finished.
   ```

3. In the MATLAB Command Window, type

   ```
   close (tg)
   xpctargetping
   ```

   For a working connection between the host PC and target PC, MATLAB displays the message

   ```
   ans=
   success
   ```

4. Type

   ```
   tg = xpc
   ```

   If the connection is resumed (Connected = Yes), then the connection is all right. If the connection is timing out consistently for a particular model, then the time out needs to be increased. See “Increasing the Time Out Value” on page 3-33.

For information on setting up the xPC Target software environment, see either “Environment Properties for Serial Communication” on page 2-11 or “Environment Properties for Network Communication” on page 2-19, and then see “Target Boot Disk” on page 2-23.

Increasing the Time Out Value

By default, if the host PC does not get a response from the target PC after downloading a target application and waiting about 5 seconds, the host PC software times out. On the other hand, the target PC responds only after downloading and initializing the target application.

Usually 5 seconds is enough time to download a target application, but in some cases it may not be sufficient. The time to download a a target application is mostly dependent on your I/O hardware. For example, thermocouple hardware takes longer to initialize. In this case, even though the target PC is fine, a false time-out is reported.

To increase the time-out value, use the following procedure:

1. In your MATLAB working directory, create a file called `xpcdltimeout.dat`.
2. In this file, put a single integer. For example, enter

   ```
   20
   ```

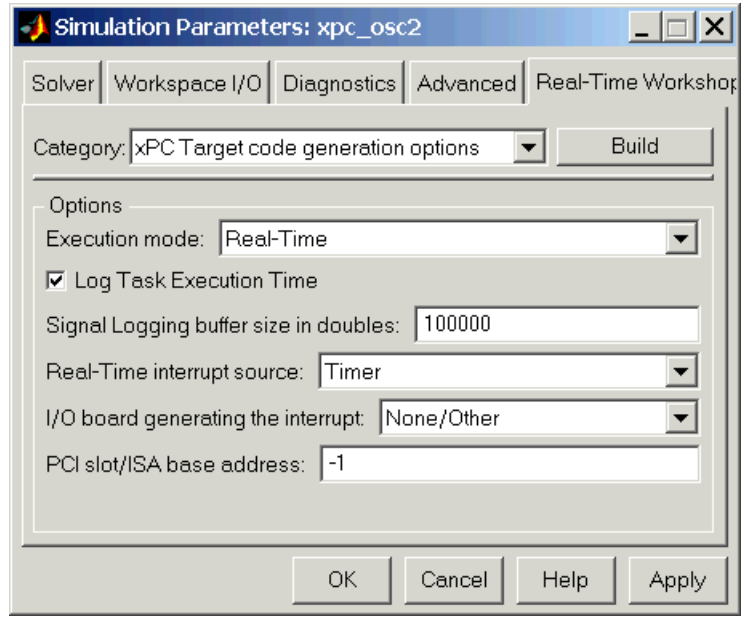
   In this case, the host PC waits for about 20 seconds before declaring that a time out has occurred. Note, it will not take 20 seconds for every download. The host PC polls the target PC about once every second, and if a response is returned, declares success. Only in the case where a download really fails will it take the full 20 seconds.

If the file `xpcdltimeout.dat` exist, it is read once before every download. To change the time out value, change the number in this file. Setting the time out to `5` is the same as the default. Note also that simply removing the file does not change the time out to the default value. xPC Target uses the last value you entered until you restart MATLAB.

xPC Target Code Generation Options

Below is a description of the fields in the xPC target code generation options page. To open this page, open the **Simulation Parameters** dialog box, chick the **Real-Time Workshop** tab, and then from the **Category** list, select `xPC Target code generation options`.

You may need to enter and select these options before you create (build) a target application. However, this is not normally necessary since the parameters have reasonable defaults.



Execution Mode — From the list select either `Real-Time` or `Freerun`.

Log Task Execution Time — Select this check box to log task execution times to the target object property `tg.TETlog`.

Signal Logging Buffer Size in Doubles — Enter the maximum number of sample points to save before wrapping. Your target application uses this buffer to store the time, states, outputs, and task execution time logs as defined in your Simulink model.

For example, the model `xpc_osc2.mdl` has 6 data items (1 time, 2 states, 2 outputs, and 1 task execution time (TET)). If you enter a buffer size of `100000`, then the target object property `tg.MaxLogSamples` is calculated as `floor(100000) / 6) = 16666`. After saving 16666 sample points, the buffer wraps and further samples overwrite the older ones.

If you select a logging buffer size larger then the available RAM on the target PC, after downloading and initializing the target application, the target PC displays a message, `ERROR: allocation of logging memory failed`. 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 rebooted.

Real-Time Interrupt Source — From the list select `Timer` or a number from `5` to `15`.

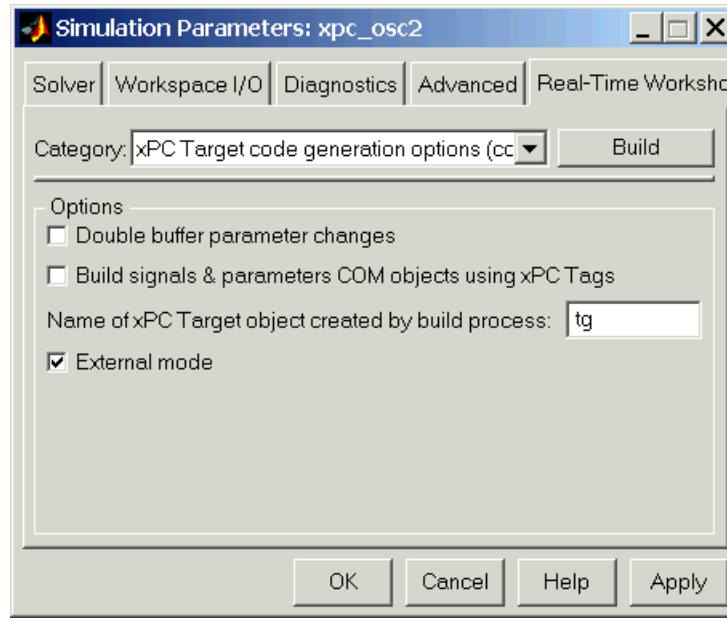
I/O board generating the interrupt — From the list, select `None/Other`, `CB_CIO-CTR05`, `CB_PCI-CTR05`, `Softing_CAN-AC2-104`, `Softing_CAN-AC2-PCI`, or `RTD_DM6804`.

PCI slot/ISA base address — Enter the slot number or base address for the I/O board generating the interrupt. The PCI slot can be either `-1` (let xPC Target determine the slot number) or of the form `[bus, slot]`. The base address is a hexadecimal number of the form `0x300`.

To determine the bus and PCI slot number of the boards in the target PC, in the MATLAB Command Window, type `getxpcpci`.

xPC Target Code Generation Options (cont.) Page

Below is a description of the fields in the xPC Target code generation options (cont.) page. To open this page, open the **Simulation Parameters** dialog box, click the **Real-Time Workshop** tab, and then from the **Category** list, select `xPC Target code generation options (cont.)`.



Double buffer parameter changes — Selecting this option changes parameter tuning so that the process of changing parameters in the target application uses a double buffer.

- When a parameter change request is received, the new value is compared to the old one. If the new value is identical to the old one, it is discarded, and if different, queued.
- At the start of execution of the next sample of the real-time task, all queued parameters are updated. This means that parameter tuning affects the task execution time (TET), and the very act of parameter tuning may cause a CPU overload error.

This approach leads to a more robust parameter tuning interface, at the cost of increased TET and a higher probability of overloads. You should normally clear this check box. It is cleared by default.

Build signals & parameters COM objects using xPC Target Tags — If you select this check box, the build process creates a model specific COM library file

```
<model_name>COMiface.dll
```

Use this file to create custom GUI interfaces with Visual Basic or other tools that can use COM objects.

Name of xPC Target object created by build process — Enter the name of the target object created by the build process. The default target object name is `tg`.

External Mode — Select this check box to activate external mode for parameter tuning from your Simulink model. This check box is selected by default.

xPC Target does not currently support signal uploading using external mode. Instead of using a Simulink Scope block, use an xPC Target Scope block to trace signals.

Running the Target Application

During the build process, xPC Target creates a target object that represents the target application running on the target PC. The target object is defined by a set of properties and associated methods. You control the target application and computer by setting the target object properties. This section includes the following topics:

- **Control with xPC Target Remote Control Tool** — Use a basic GUI interface to change target application parameters, add xPC Target Scopes, and select and log signals from your target application
- **Control with MATLAB Commands** — Start and stop the target application, view the status, change the stop time and sample time, download a new target application
- **Control with Simulink External Mode** — Connect a Simulink model to an xPC Target application for parameter tuning

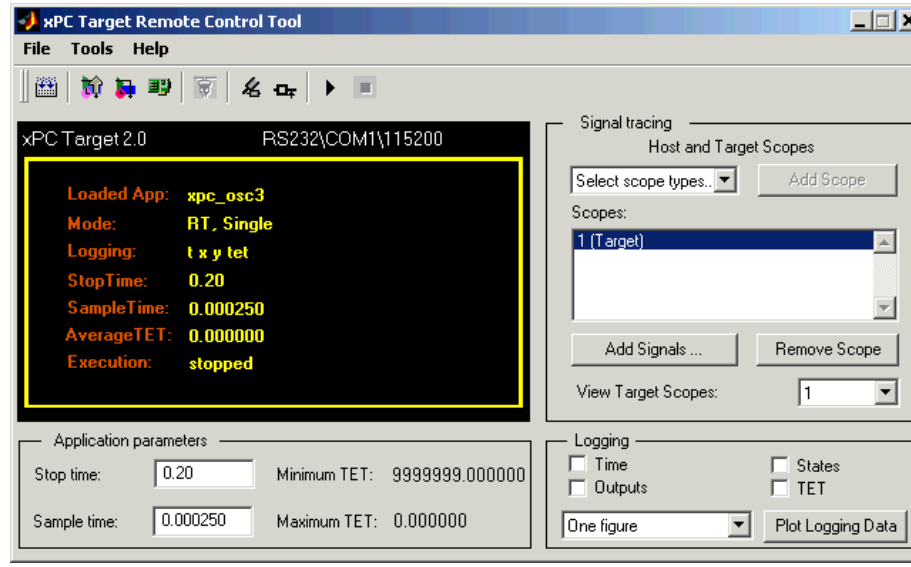
Control with xPC Target Remote Control Tool

This procedure assumes you have created an xPC Target boot disk and you have booted the target PC. See “Target Boot Disk” on page 2-23. While the xPC Target Remote Control Tool has the ability to build and download a target application, this procedure begins with a target application already downloaded to the target PC. See “xPC Target Application” on page 3-24.

1 In the MATLAB Command Window, type

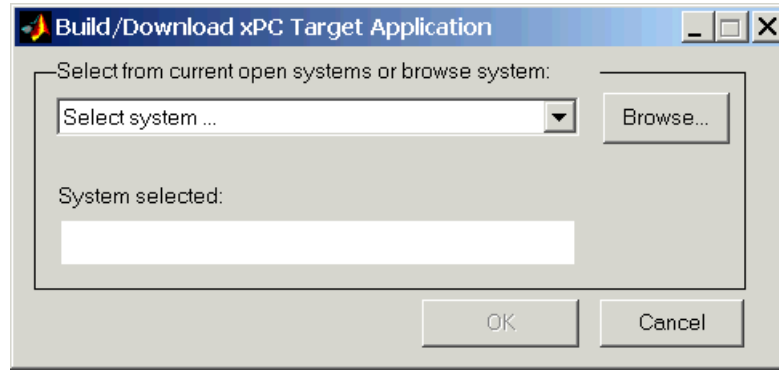
```
xpcrctool
```

The **xPC Target Remote Control Tool** window opens, connects to the target PC, and displays information about the previously loaded target application.



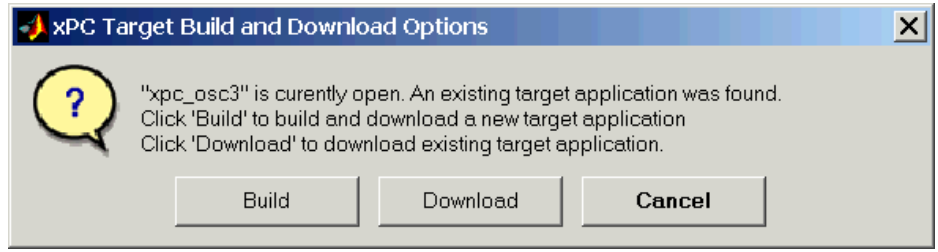
2 If you want to rebuild the target application, from the **File** menu, click **Build/Load Application**.

The **Build/Download xPC Target Application** dialog box opens.



3 From the **Current Open Systems** list, select the name of a Simulink model (`.mdl`) you have open, or click the **Browse** button and select the name of a Simulink model or a previously built target application (`.dlm`). Click **OK**.

The **Build and download Options** dialog box opens.



4 Select a download option.

- If you click **Build**, Real-Time Workshop creates a target application from the Simulink model and downloads it to the target PC.
- If you click **Download**, `xpcrctool` looks for a previously built target application for the selected Simulink model and downloads it to the target PC.

5 From the toolbar, click the start button . The target application begins running on the target PC, and then stops when it reaches the stop time.

6 In the **Stop time** box, enter a new stop time. For example, enter

```
inf
```

7 Again, click the start button. The target application now runs until you stop it.

8 From the toolbar, click the stop button .

See also “Signal Tracing with xPC Target Remote Control Tool” on page 4-3, “Signal Logging with xPC Target Remote Control Tool” on page 4-18, and “Parameter Tuning with xPC Target Remote Control Tool” on page 4-24.

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 `xpc_osc3.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`.

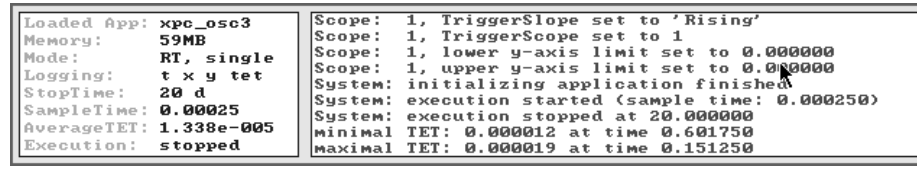
1 In the MATLAB Command Window, type

```
+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 = xpc_osc3
     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.



2 In the MATLAB Command 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 `StopTime` and `SampleTime`.

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

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

4 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 `CPUOverload` changes to `detected`. In that case, change the **Fixed step size** in the Solver pane to a larger value and rebuild the model.

Control with Simulink External Mode

Control of your xPC Target application with Simulink is limited to connecting a Simulink model to a target application through external mode, and starting the target application. Using Simulink external mode is one method to tune parameters.

After you create and download a target application to the target PC, you can run the target application. This procedure uses the Simulink model `xpc_osc2.mdl` as an example. See “Building and Downloading the Target Application” on page 3-30.

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. Simulink external mode connects your Simulink model to your target application as a simple graphical user interface.

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 where # is the actual number of tunable parameters in your model:

   ```
   ExtM: Updating # parameters
   ```

3 From the **Simulation** menu, click **Start real-time code**.

   The **StopTime** is changed to `99999999` seconds, the target application begins running, and the target PC displays the following message:

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

4 In the MATLAB Command Window, type

   ```
   tg.stop or -tg
   ```

   You cannot stop the target application from the Simulink window by clicking **Stop real-time code** from the **Simulation** menu.

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

4

Signals and Parameters

Changing parameters in your target application while it is running in real time, and checking the results by viewing signal data, are two important prototyping tasks. xPC Target includes command-line and graphical user interfaces to complete these tasks. This chapter includes the following sections:

Signal Monitoring with MATLAB

Signal monitoring is the process for acquiring signal data during a real-time run without time information. The advantage with signal monitoring is that there is no additional load on the real-time tasks. 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. This procedure uses the model `xpc_osc3.mdl` as an example, and assumes you created and downloaded the target application to the target PC:

1 To get a list of signals, 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 `xpc_osc3.mdl` are shown below.

```
ShowSignals = On
   Signals = PROP.   VALUE            BLOCK NAME
             S0      0.000000         Transfer Fcn
             S1      0.000000         Signal Generator
```

The signal names (S0, S1 . . .) are properties of a target object. See also "Signal Tracing with MATLAB" on page 4-13.

2 To get the value of a signal, in the MATLAB Command Window, type

```
tg.S1
```

MATLAB displays the value of the signal `S1`.

```
ans=
     3.731
```

Signal Tracing

Signal tracing is the process of acquiring and visualizing signals while running a target application. It allows you to acquire signal data and visualize it on the target PC or upload the signal data and visualize it on the host PC while the target application is running. This section includes the following topics:

- **Signal Tracing with xPC Target Remote Control Tool** — Use the xPC Target GUI (scope dialog boxes) to create and run scopes that display on the host PC
- **Signal Tracing with MATLAB** — Use the MATLAB Command Window to create scopes and scope objects
- **Signal Tracing with xPC Target Scope Blocks** — Use scopes of type host to trace signal data triggered by an event while your target application is running
- **Signal Tracing with a Web Browser** — Use Microsoft Explorer or Netscape Navigator to view signals

Signal tracing differs from signal logging. With signal logging you can only look at signals after a run is finished, and the entire log of the run is available. For information on signal logging, see "Signal Logging" on page 4-18.

Signal Tracing with xPC Target Remote Control Tool

Opening a xPC Target Scope window on the host or target PC allows you to view signals while running and testing your target application.

After you create and download a target application to the target PC, you can view signals. This procedure uses the Simulink model `xpc_osc4.mdl` as an example, and assumes you have created a target application and downloaded it to the target PC. See "xPC Target Application" on page 3-24.

1 In the MATLAB Command Window, type

```
xpcrctool
```

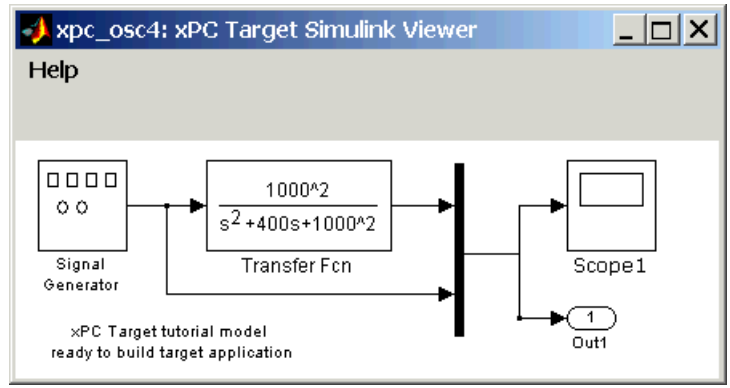
The **xPC Target Remote Control Tool** window opens and connects to the target application.

2 From the scope type list, select either `Target scope` or `Host scope`. For example, select `Target scope`, and then click **Add Scope**.

xPC Target creates a scope on the target PC, and in the **Scopes** list, displays `1(Target)`.

3 Click **Add Signals**.

The **xPC Target Simulink Viewer** window opens with a diagram of your Simulink model.

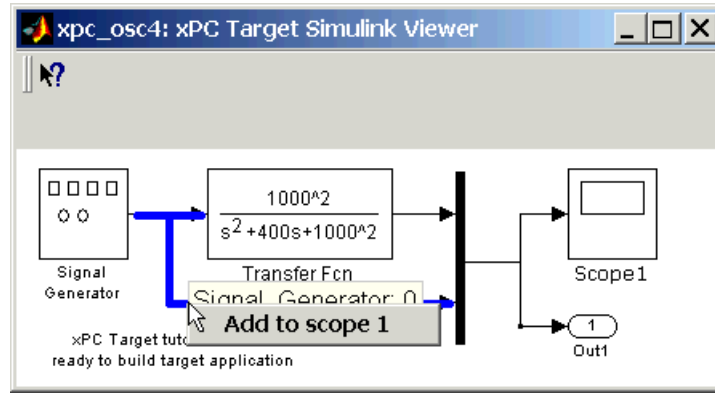


4 Move the mouse pointer over the signal lines.

- A blue line indicates that you can add this signal to a scope.
- A red line, with the message `Signal cannot be monitored`, indicates that you cannot add this signal to a scope.

5 Point to a signal line and right-click. For example, right-click over the signal from the Signal Generator.

A menu opens with a list of scopes and options to add a signal or to remove a previously added signal.



6 From the menu, click an option. For example, click **Add to scope 1**.

The signal from the Signal Generator is added to the scope on the target PC screen.

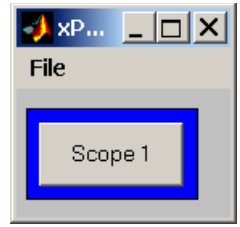
7 Point to the signal from the Transfer Function block, right-click, and select **Add to scope 1**.

8 From the toolbar, click the start button . The target application and scope begin running on the target PC, and then stop when the target application reaches the stop time.

Setting Trigger for Signals with the Target Scope Manager

1 In the **xPC Target Remote Control Tool** window, from the **Tools** menu, click **Target Scope Manager**.

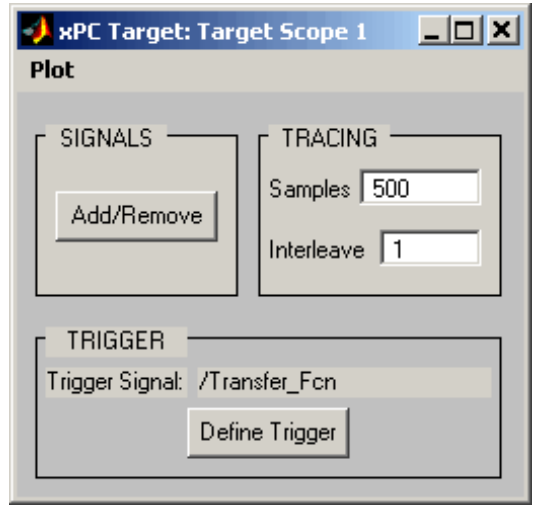
The scope manager window opens. This window is the root window of the xPC Target Scope graphical interface, and it displays the scopes that you have added.



2 Point to the **Scope 1** button and right-click. A menu opens.

3 From the menu, click **Properties**.

The **xPC Target: Target Scope 1** window opens.



4 Click **Define Trigger**. The **xPC Target: Trigger for Target Scope 1** window opens.

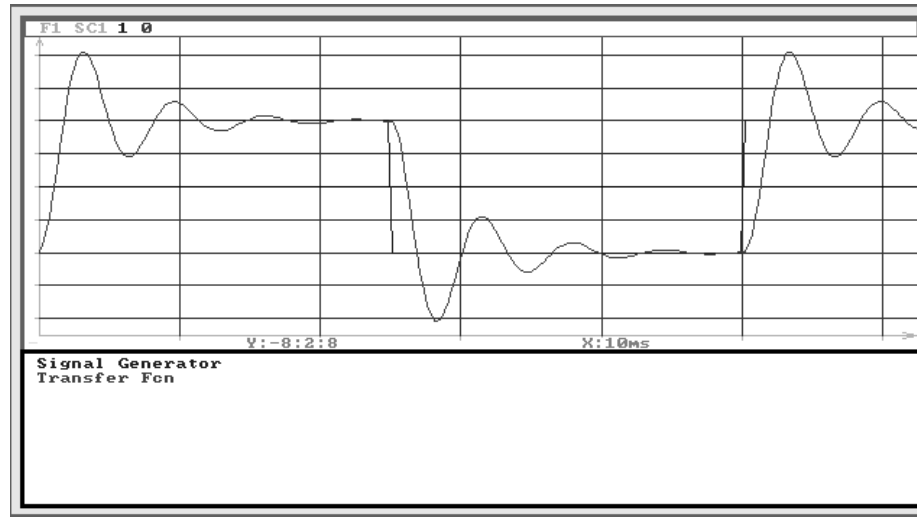
5 From the **Mode** list, select `Software`, `Signal`, or `Scope`. For example, select `Signal`.

6 From the **Slope** list, select `EITHER`, `RISING`, or `FALLING`. For example, select `RISING`.

7 From the **Signal** list, click a signal to trigger the scope. For example, click `Signal_generator`.

8 In the **xPC Target Remote Control Tool** window, from the toolbar, click the start button .

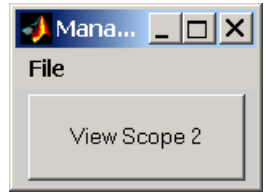
The target application and scope begin to run on the target PC, and then stop when the target application reach the stop time.



Adding and Removing Signals with the Host Scope Manager

1 In the **xPC Target Remote Control Tool** window, from the **Tools** menu, click **Host Scope Manager**.

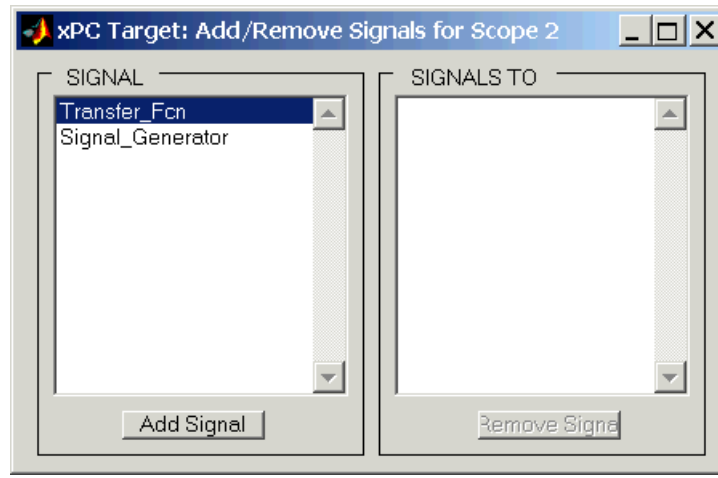
The scope manager window for scopes of type host opens.



2 Click the button for a scope. For example, click **View Scope 2**. The **xPC Target: Scope 2** window opens.

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

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

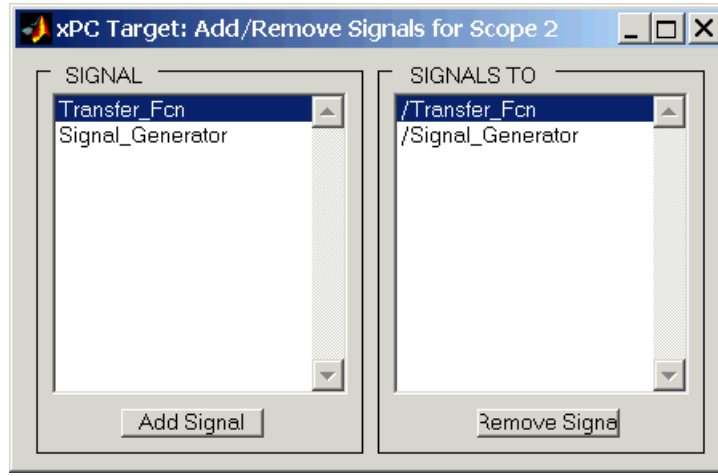


The **SIGNAL** list box displays all the available signals from the target application. The names of the signals shown correspond to the block names within the Simulink model `xpc_osc4.mdl`. The block name indicates the output signal from that block.

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

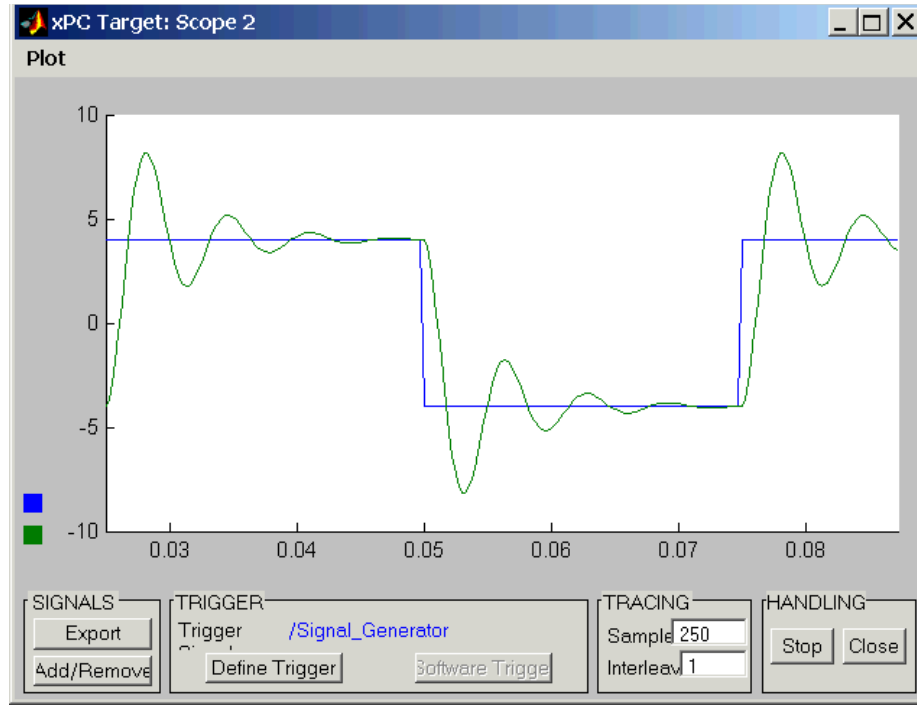
4 From the **SIGNAL** list box, click `Transfer_Fcn`, and then click the **Add Signal** button. Similarly, add the `Signal_Generator` signal.

Changes to the **Add/Remove Signals** dialog box are shown below. You can remove the signals to trace by clicking the block name in the **SIGNALS TO** 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.

5 In the **xPC Target Remote Control Tool** window, from the toolbar, click the start button . The target application and scope begin to run on the target PC, and then stop when they reach the stop time.



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

Saving Data from a Scope of Type Host to the MATLAB Workspace

While a target application is running, data is saved in an xPC Target Scope buffer. You can export the buffer to MATLAB. Exporting data with the **xPC Target Scope** window does not require you to add Outport blocks to your Simulink model, nor does it require you to activate the logging of signals. You can select the signals to collect, and you can capture unexpected outputs during a run.

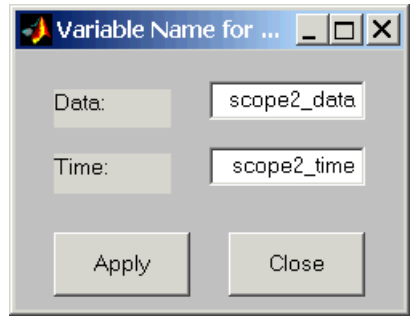
After you complete a run with a scope, you can move data from the last data package collected to the MATLAB workspace.

1 In the **xPC Target Remote Control Tool** window, from the **Tools** menu, click **Host Scope Manager**.

   The scope manager window and the **xPC Target Scope** window open.

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

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



3 In the **Data** and **Time** text boxes, enter the names of the MATLAB variables to save data from a trace. Click the **Apply** button, and then click the **Close** button. The default name for the time vector is `scope`*n*`_time`, and the default name for the signal vector is `scope`*n*`_data` where *n* is the scope number.

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

5 In the MATLAB Command Window, type

```
whos
```

   MATLAB displays a list of variables and their descriptions. For example:

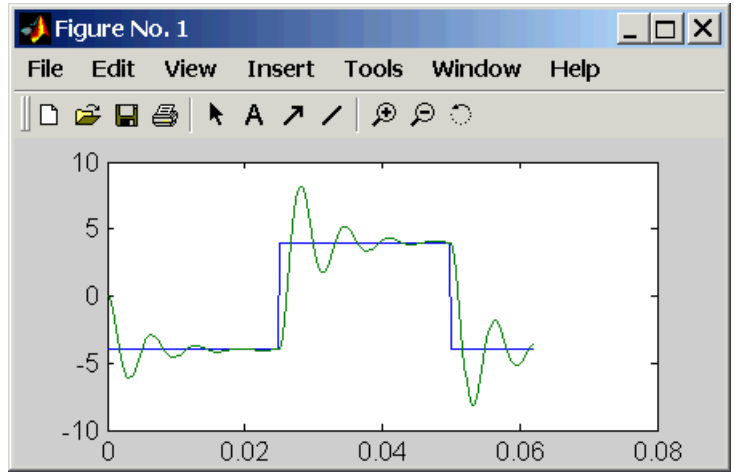
```
Name              Size         Bytes  Class
  scope2_data     250x2           4000  double array
  scope2_time     250x1           2000  double array
  tg                1x1          10174 xpc object
Grand total is 1644 elements using 16174 bytes
```

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

6 Type

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

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



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 whether 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 MATLAB

Creating a scope object allows you to select and view signals using the scope methods. This section describes how to signal trace using xPC Target functions instead of using the xPC Target graphical user 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 this 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 signals, type either

```
set(tg, 'ShowSignals', 'on')
```

or type

```
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. For more information, see “Signal Monitoring with MATLAB” on page 4-2.

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
   TriggerSample        = -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

```
sc1.addsignal ([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. 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'
```

Signal Tracing with xPC Target Scope Blocks

Use scopes of type host to log signal data triggered by an event while your target application is running.

Scope of Type Host

For a scope of type host, the scope acquires the first `N` samples into a buffer. You can retrieve this buffer into the scope object property (`sc.Data`). The scope then stops and waits for you to manually restart the scope.

The number of samples `N` to log after triggering an event is equal to the value you entered in **Number of Samples**.

Select the type of event in the **Block Parameters: Scope (xPC Target)** dialog box by setting **Trigger Mode** to `Signal Triggering`, `Software Triggering`, or `Scope Triggering`.

Scope of Type Target

For a scope of type target, logged data (`sc.Data`, `sc.Time`, and `sc.StartTime`) is not accessible over the command-line interface on the host PC. This is because the scope object status (`sc.Status`) is never set to `Finished`. Once the scope completes one data cycle (time to collect the Number of Samples), the scope engine automatically restarts the scope.If you create a scope object (for example, sc = getscopes(tg,1) for a scope of type target, and then try to get the logged data by typing `sc.Data`, you will get an error message.

```
Scope # 1 is of type 'Target'! Property Data is not accessible.
```

If you want the same data for the same signals on the host PC while the data is displayed on the target PC, you need to define a second scope object with type host. Then you need to synchronize the acquisition of the two scope objects by setting **TriggerMode** for the second scope to `'Scope'`.

Signal Tracing with a Web Browser

The Web browser interface allows you to visualize data using an interactive graphical user 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 browser loads the **Scopes List** page 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. You can add a new scope, remove existing scopes, and control all aspects of a scope from this page.

   To create a scope of type host, use the drop-down list next to the **Add Scope** button to select `Host`. This item is set to `Target` by default.

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 necessary, the Web interface stops the scope automatically and then restarts it when the changes are made. It does not restart the scope if the state was originally stopped.

When a host scope is stopped (**Scope State** is set to `Interrupted`) or finishes one cycle of acquisition (**Scope State** is set to `Finished`), a button called **Get Data** appears on the page. If you click this button, the scope data is retrieved in comma separated variable (CSV) format. The signals in the scope are spread across columns, and each row corresponds to one sample of acquisition. The first column always corresponds to the time each sample was acquired.

Note If **Scope State** is set to `Interrupted`, the scope was stopped before it could complete a full cycle of acquisition. Even in this case, the number of rows in the CSV data will correspond to a full cycle. The last few rows (for which data was not acquired) will be set to 0.

Signal Logging

Signal logging is the process for acquiring signal data during a real-time run, stopping the target application, and then transferring the data to the host PC for analysis. You can plot and analyze the data, and later save it to a disk. This section includes the following topics:

- **Signal Logging with xPC Target Remote Control Tool** — Use an xPC Target Scope block in your Simulink model to log signal data triggered by an event.
- **Signal Logging with MATLAB** — Use Outport blocks in your Simulink model to log data to a target object in the MATLAB workspace.
- **Signal Logging with a Web Browser** — Use Microsoft Internet Explorer or Netscape Navigator to log data to a text file.

Signal Logging with xPC Target Remote Control Tool

You 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 and model parameters.

This procedure uses the model `xpc_osc3.mdl` as an example and assumes you have created a target application and downloaded it to the target PC. See "xPC Target Application" on page 3-24.

Note To use the xPC Target Remote Control Tool for signal logging, you need to add an Outport block to your Simulink model, and you need to activate logging on the I/O Workspace pane in the **Simulation Parameters** dialog box.

---

1 In the MATLAB Command Window, type

```
xpcrctool
```

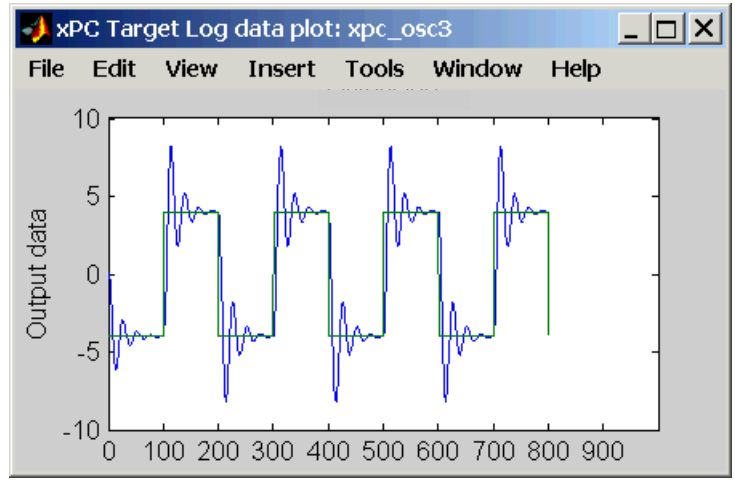
The **xPC Target Remote Control Tool** window opens, connects to the target PC, and displays information about a previously loaded target application.

2 Select the **Outputs** check box and the **Plot vs Sample** option.

3 From the toolbar, click the start button . The target application begins running on the target PC, and then stops when it reaches the stop time.

4 Click the **Plot Logging Data** button.

MATLAB opens a figure window and draws a plot of the outputs.



Note Logged data is available only when the target application is not running.

Signal Logging with MATLAB

You 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 and model parameters.

Time, states and outputs — Logging the output signals is possible only if you add, before the build process, outport blocks to your Simulink model, and in the **I/O-Workspace** pane select the **Save to workspace** check boxes. See "Entering Parameters for Outport Blocks" on page 3-7.

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**. This check box is selected by default. See "Adding an xPC Target Scope Block" on page 3-11.

A scope of any type copies the last `N` samples from the log buffer to the target object logs (`tg.TimeLog`, `tg.OutputLog`, `tg.State` and `Log`, `tg.TETLog`). The number of samples `N` for a signal is equal to the value you entered in **Signal Logging Buffer Size in Doubles** and divided by the number of logged signals (1 time, 1 TET, Outputs, States)

After you run a target application, you can plot the state and output signals. This procedure uses the Simulink model `xpc_osc3.mdl` as an example, and assumes you have created and downloaded the target application for that model.

1 In the MATLAB Command 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`.

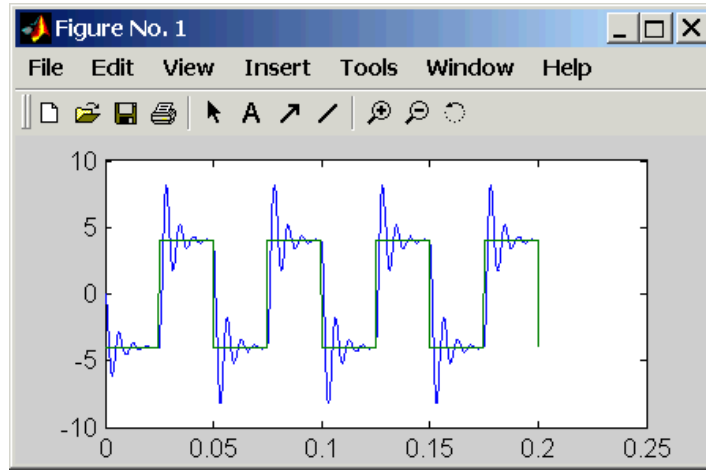
The outputs are the signals connected to Simulink outport blocks. The model `xpcosc.mdl` has just one outport block labeled **1** and there are two states. This outport block shows the signals leaving the blocks labeled Integrator1 and Signal Generator.

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

```
plot(tg.TimeLog,tg.Outputlog)
```

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 5-22 in the xPC Target User Guide documentation.

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 “Simulating the Model from MATLAB” on page 3-22.

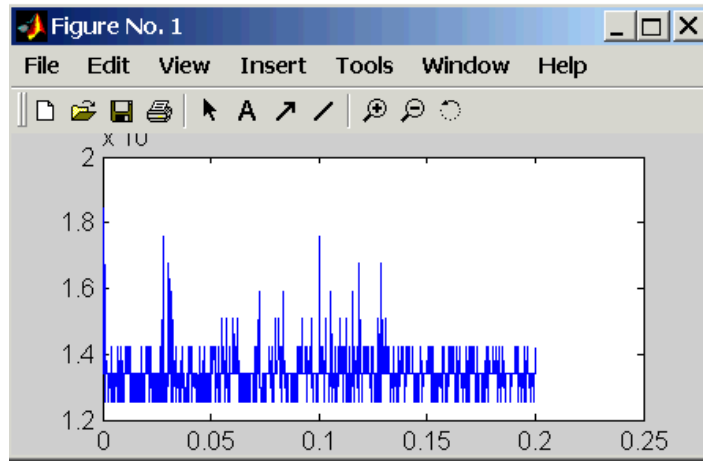


3 In the MATLAB Command Window, type

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

Values for the task execution time (TET) log is uploaded to the host PC from the target PC. If you want to upload part of the logs, see the target object method `getlog` on page 5-22 in the xPC Target User Guide documentation.

The plot shown below is the result of a real-time run.



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.

4 In the MATLAB Command Window, type either

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

MATLAB displays the following information about the average task execution time.

```
ans =
0.000013
```

In the example above, the minimum TET was 13 µs, the maximum TET 18 µs, and the average TET 13µs. This means that the real-time task has taken about 5% of the CPU performance (Average TET of 13 µs / Sample time of 250 µs).

Signal Logging with a Web Browser

When you stop the model execution, another section appears that enables you to download logging data. This data is in CSV format. This format can be read by most spreadsheet programs and also by MATLAB using the `csvread` command.

This section will appear only if you have enabled data logging, and buttons appear only for those logs (States, Output, and TET) that are enabled. If time logging is enabled, the first column of the CSV file is the time at which data (state values, output values, and TET values) was acquired. If time logging is not enabled, only the data is in the CSV file without time information.

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 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 Parameters for Outport Blocks” on page 3-7.

Task execution time — Logging 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**. This check box is selected by default. See “Entering Parameters for an xPC Target Scope Block” on page 3-15.

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 xPC Target Remote Control Tool** — Use the xPC Target Remote Control Tool to change block parameters in your target application
- **Parameter Tuning with MATLAB** — Use the MATLAB Command Window and target objects in your MATLAB workspace to change target application parameters
- **Parameter Tuning with Simulink External Mode** — Connect your Simulink model to your target application, and change target application parameters by changing Simulink block parameters

---

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

- **Parameter Tuning with a Web Browser** — Connect your target application to a Web browser with the target application running on a target PC connected to a network

Parameter Tuning with xPC Target Remote Control Tool

You can change parameters in your target application using the xPC Target Simulink Viewer whether the target application is stopped or running.

After you create and download a target application to the target PC, you can view signals. This procedure uses the Simulink model `xpc_osc4.mdl` as an example, and assumes that you have created a target application and downloaded it to the target PC. See “xPC Target Application” on page 3-24.

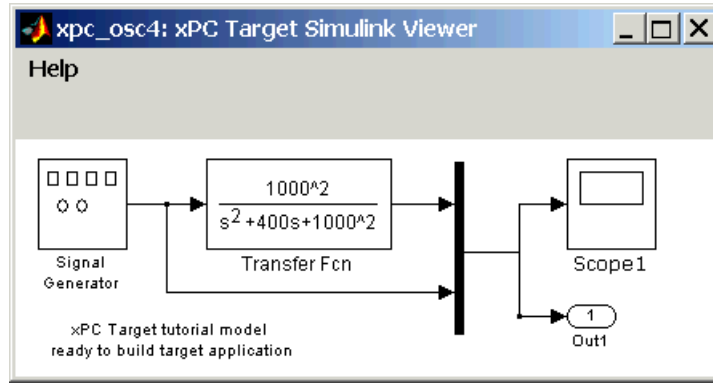
1 In the MATLAB Command Window, type

```
xpcrctool
```

The **xPC Target Remote Control Tool** window opens and connects to the target application.

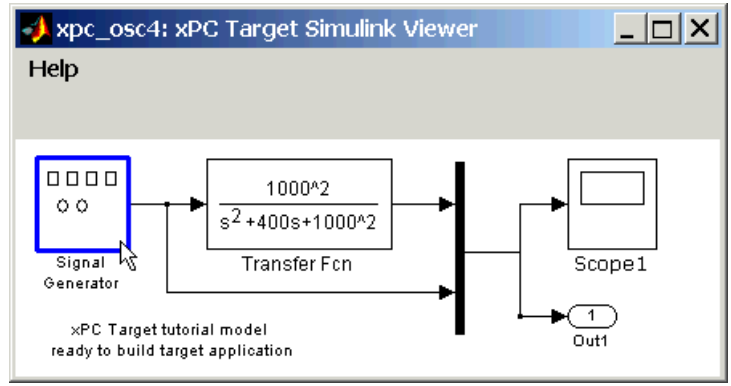
2 From the **Tools** menu, click **Tune Parameters**.

The **xPC Target Simulink Viewer** window opens with a diagram of your Simulink model.



3 Move the mouse pointer over the blocks.

- A blue highlight around a block indicates that this block contains tunable parameters.
- A red highlight around a block indicates that there are no tunable parameters.



4 Point to a block and right-click. For example, right-click over the Signal Generator block.

   The **Block Parameters** dialog box opens.

5 Enter new parameters, and then click OK.

   The new parameter values are downloaded to the target application. If the target application is running, you can see the effect of the change.

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

Parameter Tuning with MATLAB

You use the MATLAB functions to change block parameters. With these functions you do not need to set Simulink to 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 target application without rebuilding the Simulink model.

After you download a target application to the target PC, you can change block parameters using xPC Target 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 Command 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 properties for the target object.

```
ShowParameters = On
Parameters= PROP VALUE     PARAMETER NAME    BLOCK NAME
            P0   0.000000  InitialCondition  Integrator1
            P1   4.000000  Amplitude         Signal Generator
            P2   20.000000 Frequency         Signal Generator
            P3   1000000.0 Gain              Gain
            P4   0.000000  InitialCondition  Integrator
            P5   400.00000 GainGain1
            P6   1000000.0 GainGain2
```

The parameter names (P0, P1, . . .P6) are 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 Command Window, type either

```
-tg or tg.stop 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.

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

After you 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 where # is the actual number of tunable parameters in your model.

```
ExtM: Updating # parameters
```

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

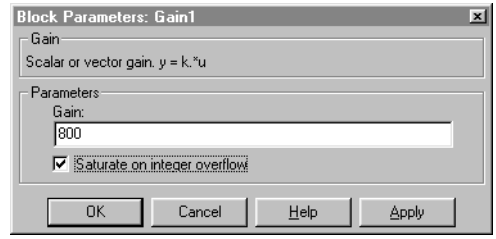
```
+tg or tg.start or start(tg)
```

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

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

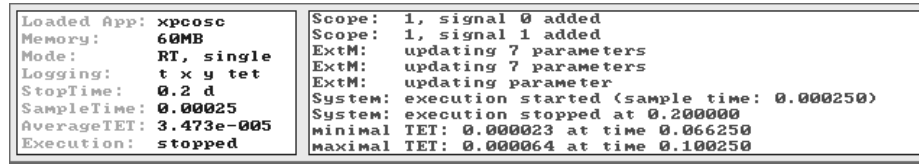
4 From the Simulation block diagram, click the block labeled **Gain1**.

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.



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
```

Parameter Tuning with a Web Browser

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 browser loads the **Parameter List** page 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 that 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.

Index

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X