

# xPC TargetBox<sup>®</sup>

**For Use with Real-Time Workshop<sup>®</sup>**

- Modeling
- Simulation
- Implementation

User's Guide

*Version 1*



## How to Contact The MathWorks



[www.mathworks.com](http://www.mathworks.com)  
[comp.soft-sys.matlab](mailto:comp.soft-sys.matlab)  
[www.mathworks.com/contact\\_TS.html](http://www.mathworks.com/contact_TS.html)

Web  
Newsgroup  
Technical Support



[suggest@mathworks.com](mailto:suggest@mathworks.com)  
[bugs@mathworks.com](mailto:bugs@mathworks.com)  
[doc@mathworks.com](mailto:doc@mathworks.com)  
[service@mathworks.com](mailto:service@mathworks.com)  
[info@mathworks.com](mailto:info@mathworks.com)

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



508-647-7000 (Phone)



508-647-7001 (Fax)



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

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

*xPC TargetBox User's Guide*

© COPYRIGHT 2002–2006 by The MathWorks, Inc.

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

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

### Trademarks

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

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

### Patents

The MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

### **Revision History**

November 2002	First printing	New for xPC Target 2.0 software (Release 13)
September 2003	Second Printing	Revised for xPC Target 2.0.1 software (Release 13SP1)
January 2004	Third printing	Revised for xPC Target 2.5 software (Release 14)
August 2004	Online only	Revised for xPC Target 2.6 software (Release 14+)
October 2004	Fourth printing	Revised for xPC Target 2.6.1 software (Release 14SP1)
November 2004	Online only	Revised for xPC Target 2.7 software (Release 14SP1+)
March 2005	Online only	Revised for xPC Target 2.7.2 software (Release 14SP2)
September 2005	Online only	Revised for xPC Target 2.8 software (Release 14SP3)
March 2006	Online only	Revised for xPC Target 2.9 software (Release 2006a)
May 2006	Online only	Revised for xPC Target 3.0 software (Release 2006a+)
September 2006	Online only	Revised for xPC Target 3.1 software (Release 2006b)



## **Limited Warranty from MPL**

WARRANTY: MPL WARRANTS THAT THE PRODUCTS DELIVERED HEREUNDER SHALL BE FREE FROM DEFECTS IN WORKMANSHIP AND MATERIAL FOR A PERIOD OF TWELVE (12) MONTHS FROM DATE OF DELIVERY TO THE BUYER, INCLUDING COMPONENT PARTS OF PRODUCTS SOLD AS SPARE, REPLACEMENT, MAINTENANCE OR STORAGE PARTS, WHICH ARE ALSO WARRANTED FOR TWELVE (12) MONTHS FROM DATE OF DELIVERY, PROVIDED, HOWEVER, IN EITHER CASE, THAT NOTICE OF ANY SUCH DEFECT IS PROVIDED TO MATHWORKS OR MPL WITHIN THIRTY (30) DAYS OF ITS DISCOVERY BY THE BUYER. NEITHER MATHWORKS NOR MPL MAKE ANY OTHER WARRANTY OF ANY KIND TO BUYER, AND HEREBY EXPRESSLY DISCLAIMS ANY SUCH OTHER WARRANTY, WHETHER EXPRESS OR IMPLIED, IN FACT OR BY LAW, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT OR WARRANTY OF ANY KIND INCORPORATED OR REFERENCED IN BUYER'S SPECIFICATIONS OR PURCHASE ORDER, OR ANY OTHER WARRANTY ARISING BY STATUTE, OR OTHERWISE IN LAW, OR FROM A COURSE OF DEALINGS OR USE OF TRADE, ALL OF WHICH ARE HEREBY DISCLAIMED.

EXCLUSIVE REMEDY: IN ANY EVENT, THE BUYER'S EXCLUSIVE REMEDY HEREUNDER IS LIMITED TO THE FURNISHING OF REPLACEMENT PARTS ON AN EXCHANGE BASIS, OR, AT THE OPTION OF MPL, TO THE REPAIR OR REPLACEMENT OF DEFECTIVE PRODUCTS OR COMPONENT PARTS AT ITS PLANT, BUT IN EITHER CASE ONLY SO LONG AS AN EXAMINATION WITHIN THE PERIOD OF WARRANTY REVEALS THE PARTS TO BE DEFECTIVE, AND IN ALL CASES ALL COSTS OF SHIPPING AND PACKAGING SHALL BE BORNE BY THE BUYER.

IN ADDITION, AND NOTWITHSTANDING ANYTHING HEREIN TO THE CONTRARY, NEITHER MATHWORKS NOR MPL SHALL INCUR ANY OBLIGATION HEREUNDER WITH RESPECT TO PRODUCTS WHICH ARE MODIFIED IN ANY WAY BY THE BUYER WITHOUT THE PRIOR WRITTEN CONSENT OF MPL, AND IN NO EVENT SHALL MATHWORKS OR MPL INCUR ANY OBLIGATION TO REPAIR OR REPLACE PRODUCTS OR COMPONENT PARTS WHICH ARE

DETERMINED BY MPL OR SELLER, IN ITS SOLE DISCRETION, TO BE DEFECTIVE DUE TO BUYER MISUSE, USE OF UNAUTHORIZED REPAIR PARTS OR UNAUTHORIZED THIRD-PARTY SERVICE, OR BECAUSE OF A USE NOT IN ACCORDANCE WITH SPECIFIC MPL OR SELLER PRODUCT OPERATION AND MAINTENANCE INSTRUCTIONS. CUSTOMER ACCEPTS AND ACKNOWLEDGES THAT THE FOREGOING ALLOCATION OF RISK IS REFLECTED IN THE PURCHASE PRICE.

LIMITATION OF LIABILITY: IN NO EVENT SHALL MATHWORKS OR MPL BE LIABLE FOR SPECIAL, EXEMPLARY, INDIRECT, PUNITIVE, CONSEQUENTIAL OR INCIDENTAL DAMAGES INCLUDING BUT NOT LIMITED TO LOSS OF PROFITS, LOSS OF BUSINESS OR GOODWILL, OR LOSS OF USE FOR ANY BREACH OF THESE TERMS AND CONDITIONS OR ARISING UNDER CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, WARRANTY OR ANY OTHER THEORY OF LIABILITY OR ANY LIABILITY CLAIM OF ANY THIRD PARTY EVEN IF MATHWORKS OR MPL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

FOR ANY BREACH OF THESE TERMS AND CONDITIONS, MATHWORKS' AND MPL'S MAXIMUM LIABILITY SHALL NOT IN ANY EVENT EXCEED THE TOTAL PRICE OF THE PRODUCTS ORDERED BY CUSTOMER.

NEITHER MATHWORKS NOR MPL SHALL BE LIABLE IF IT IS UNABLE TO PERFORM ANY OF ITS OBLIGATIONS CONTAINED IN THESE TERMS AND CONDITIONS DUE TO ANY INDUSTRIAL DISPUTE, WAR, FLOOD, EXPLOSION, ACT OF GOD OR ANY OTHER EVENT BEYOND THE REASONABLE CONTROL OF MATHWORKS OR MPL.

**Repairing an xPC TargetBox Under Warranty.** Your first point of contact for all xPC Target and xPC TargetBox questions is The MathWorks. See <http://www.mathworks.com/support/product/XB/> for more information.

If a MathWorks Technical Support Representative determines that your xPC TargetBox hardware needs repair, follow the steps below:

- 1 A MathWorks Customer Service Representative will give you a Return Material Authorization (RMA) number and provide you with shipping documentation and instructions. The MathWorks will also inform MPL of your problem.

- 2** Using the shipping instructions provided by The MathWorks, package and return your xPC TargetBox to MPL. For information on correct packaging, refer to “Unpacking the Shipping Case” on page 2-2.
- 3** MPL will repair your xPC TargetBox and return it to you within the shortest possible time.

**xPC TargetBox Certification.** The xPC TargetBox has received the following certifications:

- FCC Part 15, Class A Emissions Testing (US)
- EN 55022 (Europe)
- VCCI (Japan)
- ICES-033 (Canada)
- AS/NZS 3548 (Australia/New Zealand)





## Getting Started

### 1

<b>What Is xPC TargetBox?</b> .....	<b>1-2</b>
Support for Development Process .....	<b>1-2</b>
Additional Boot Modes .....	<b>1-3</b>
<b>Quick Start Guide</b> .....	<b>1-4</b>
<b>Features of xPC TargetBox</b> .....	<b>1-6</b>
Hardware .....	<b>1-6</b>
Software .....	<b>1-8</b>
xPC TargetBox Boot Modes .....	<b>1-9</b>
Communication .....	<b>1-10</b>
I/O Options .....	<b>1-11</b>
Expected Background .....	<b>1-12</b>
<b>System Requirements</b> .....	<b>1-13</b>
Software Requirements for a Host PC .....	<b>1-14</b>
Hardware Requirements for a Host PC .....	<b>1-15</b>
xPC TargetBox Software .....	<b>1-15</b>
xPC TargetBox Hardware .....	<b>1-16</b>
xPC TargetBox I/O Options .....	<b>1-18</b>

## Initial Loopback Test

### 2

<b>Installation for Loopback Testing</b> .....	<b>2-2</b>
Unpacking the Shipping Case .....	<b>2-2</b>
Running the Self-Test Without a Monitor .....	<b>2-6</b>
Running the Self-Test With a Monitor .....	<b>2-11</b>
Troubleshooting the Loopback Self-Test .....	<b>2-15</b>

<b>Connecting Hardware to an xPC TargetBox</b> .....	<b>2-18</b>
Introduction to I/O Options .....	<b>2-18</b>
Configuration Label for I/O Options .....	<b>2-19</b>
Planning I/O Hardware Connections .....	<b>2-27</b>
Connecting Hardware to Screw Terminal Boards .....	<b>2-28</b>

## Regular Use

# 3

<b>xPC Target Software Installation</b> .....	<b>3-3</b>
xPC Target Software .....	<b>3-3</b>
Activating the Additional xPC TargetBox Boot Modes ....	<b>3-4</b>
Configuring xPC TargetBox Host PC .....	<b>3-4</b>
xPC Target Embedded Option Software .....	<b>3-6</b>
<b>xPC TargetBox Hardware Installation</b> .....	<b>3-7</b>
Removing Test Dongles .....	<b>3-7</b>
Connecting the External Floppy Disk Drive .....	<b>3-8</b>
Connecting Additional Peripherals .....	<b>3-11</b>
<b>Host PC to xPC TargetBox Communication</b> .....	<b>3-14</b>
Hardware for Serial Communication .....	<b>3-14</b>
Environment Properties for Serial Communication .....	<b>3-14</b>
Hardware for Network Communication .....	<b>3-17</b>
Environment Properties for Network Communication ....	<b>3-17</b>
<b>xPC TargetBox Test</b> .....	<b>3-22</b>
Creating a Target Boot Disk .....	<b>3-22</b>
Booting an xPC TargetBox .....	<b>3-24</b>
Testing the Installation .....	<b>3-27</b>
Test 1, Ping Target System Standard Ping .....	<b>3-29</b>
Test 2, Ping Target System xPC Target Ping .....	<b>3-30</b>
Test 3, Reboot Target Using Direct Call .....	<b>3-31</b>
Test 4, Build and Download Application .....	<b>3-31</b>

4

<b>xPC TargetBox Library</b> .....	4-2
Drivers in the xPC TargetBox Library .....	4-2
Using the Panel LEDs .....	4-4
Using the Watchdog Timer .....	4-5
<b>DOSLoader Mode</b> .....	4-6
Updating Environment Properties and Creating a Boot Disk .....	4-7
Copying the Kernel to Flash Memory .....	4-8
Creating a Target Application .....	4-10
<b>StandAlone Mode</b> .....	4-12
Updating Environment Properties .....	4-12
Creating a Kernel/Target Application .....	4-13
<b>FTP File Transfer</b> .....	4-15
Directories and Files on the xPC TargetBox .....	4-15
Booting xPC TargetBox with FTP Server .....	4-17
Using FTP Commands .....	4-18
Copying Files with DOS and FTP .....	4-18
<b>Booting xPC TargetBox to DOS</b> .....	4-22

**xPC TargetBox I/O Options Reference**

5

<b>Introduction to I/O Options</b> .....	5-3
Pin Layout and Screw Terminal Boards .....	5-3
Making Your Own Terminal Boards .....	5-4
<b>Loopback Testing of I/O Options</b> .....	5-5
Loopback Testing Process .....	5-5
Determining the Success of a Loopback Test .....	5-6
Uses for Loopback Testing .....	5-6
Creating a Simulink Model for Loopback Testing .....	5-7

Running the Target Application for Specific I/O Option	
Testing .....	5-8
<b>xPC TargetBox IO 301 .....</b>	<b>5-10</b>
Wiring for IO 301 Test Dongles .....	5-10
Pin Numbering for Connectors and Screw Terminal	
Boards .....	5-11
Pin Layout IO 301 (1 of 2 Connectors) .....	5-11
Pin Layout IO 301 (2 of 2 Connectors) .....	5-13
Testing Model IO 301 .....	5-14
Diamond-MM-32-AT (IO 301) Driver Blocks .....	5-16
<b>xPC TargetBox IO 302 .....</b>	<b>5-27</b>
Wiring for IO 302 Test Dongle .....	5-27
Pin Layout IO 302 .....	5-27
Testing Model IO 302 .....	5-29
Ruby-MM-1612 (IO 302) Driver Blocks .....	5-30
<b>xPC TargetBox IO 303 .....</b>	<b>5-35</b>
Wiring for IO 303 Test Dongle .....	5-35
Pin Layout IO 303 .....	5-35
Testing Model IO 303 .....	5-37
Ruby-MM-416 (IO 303) Driver Blocks .....	5-38
<b>xPC TargetBox IO 304 .....</b>	<b>5-43</b>
Wiring for IO 304 Test Dongle .....	5-43
Pin Layout IO 304 (Both Connectors) .....	5-43
Testing Model IO 304 .....	5-45
Onyx-MM (IO 304) Driver Blocks .....	5-46
<b>xPC TargetBox IO 305 .....</b>	<b>5-50</b>
Wiring for IO 305 Test Dongle .....	5-50
Pin Layout IO 305 .....	5-50
Testing Model IO 305 .....	5-52
Quartz-MM-10 (IO 305) Driver Blocks .....	5-53
<b>xPC TargetBox IO 306 .....</b>	<b>5-64</b>
Wiring for IO 306 Test Dongle .....	5-64
Pin Layout IO 306 (1 of 2 Connectors) .....	5-64
Pin Layout IO 306 (2 of 2 Connectors) .....	5-66
Pin Layout and Ports .....	5-68

Testing Model IO 306 .....	<b>5-68</b>
DM6814 (IO 306) Driver Blocks .....	<b>5-70</b>
<b>xPC TargetBox IO 308 .....</b>	<b>5-74</b>
Wiring for IO 308 Test Dongle .....	<b>5-74</b>
Pin Layout IO 308 .....	<b>5-74</b>
Testing Model IO 308 .....	<b>5-75</b>
CAN-AC2-104 Driver Blocks (IO 308) .....	<b>5-76</b>
CAN-AC2-104 FIFO Driver Blocks (IO 308) .....	<b>5-86</b>
<b>xPC TargetBox IO 309 .....</b>	<b>5-106</b>
Wiring for IO 309 Test Dongle .....	<b>5-106</b>
Pin Layout IO 309 .....	<b>5-106</b>
Testing Model IO 309 .....	<b>5-108</b>
MPL PATI (IO 309) Driver Blocks .....	<b>5-109</b>

## **Index**





# Getting Started

---

This chapter describes the features of xPC TargetBox®. It supports version 1 of the xPC TargetBox hardware. The chapter also describes the xPC Target software environment. These descriptions will help you develop a process for working with these products together. This chapter includes the following sections:

What Is xPC TargetBox? (p. 1-2)	Brief description of an xPC TargetBox and the additional xPC Target boot modes
Quick Start Guide (p. 1-4)	Process for getting your xPC TargetBox up and running quickly
Features of xPC TargetBox (p. 1-6)	Describes the flexible software and hardware environment to prototype, test, and deploy real-time systems using PC-compatible hardware
System Requirements (p. 1-13)	Describes system requirements for the host PC and the system specifications of an xPC TargetBox

## What Is xPC TargetBox?

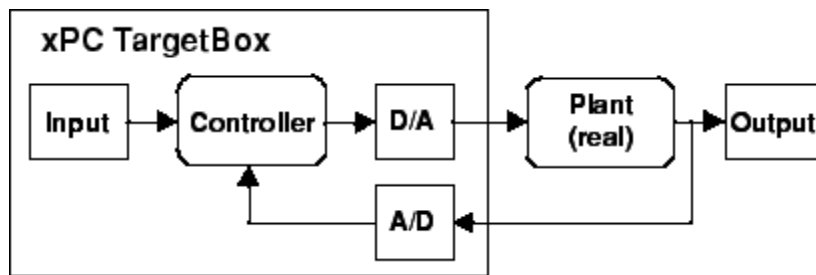
xPC Target provides the capability for prototyping, testing, and deploying real-time systems using standard PC hardware. It is an environment that uses a target PC, separate from a host PC, for running real-time applications.

xPC TargetBox is an industrial target PC. It is optimized for executing real-time code generated with xPC Target, Real-Time Workshop®, and a C/C++ compiler. With additional I/O options, you can connect to a hardware environment using analog input (A/D), analog output (D/A), digital I/O, counter/timers, encoders, interrupts, and a CAN field bus.

### Support for Development Process

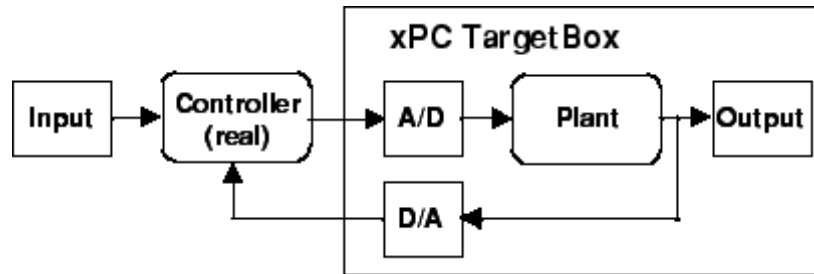
You can use an xPC TargetBox from the beginning stages of design to final deployment:

- **Rapid prototyping** — Prototype a controller running in real time on an xPC TargetBox connected to a real plant, and validate your design in real time without the need for custom target hardware.





- **Hardware-in-the-loop simulation** — Test a real controller connected to a simulated plant running in real time on an xPC TargetBox.



- **Limited deployment** — Using stand-alone mode, connect an xPC TargetBox to your plant, and run the control software without a connection to a host PC.

## Additional Boot Modes

You can generate executable code from your Simulink® and Stateflow® models on a host PC, using xPC Target, Real-Time Workshop, and a C/C++ compiler. With xPC Target and an xPC TargetBox you can use an external floppy disk drive or network connection to configure your xPC TargetBox to

- **Boot from flash disk** — Transfer the xPC Target kernel to the flash disk, boot the xPC TargetBox with the kernel, and then, from the host PC, download and run a real-time application.
- **Run stand-alone applications** — Transfer the xPC Target kernel and stand-alone application to the flash disk, disconnect the host PC, and then boot and run the real-time application.

## Quick Start Guide

xPC TargetBox is an industrial PC that lets you run target applications generated by Real-Time Workshop and xPC Target. Use the following process to get your xPC TargetBox up and running quickly:

- 1** Unpack the xPC TargetBox. When you receive your xPC TargetBox, it is delivered in a rugged shipping case. Place the case on a table with the top of the shipping case facing up, and push the lower end of the latches to open the case.

You will see the xPC TargetBox, a floppy disk drive, screw terminal boards, the AC adapter, and an AC power cord.

See “Unpacking the Shipping Case” on page 2-2.

- 2** Take out the xPC TargetBox from the shipping case.

Depending on the I/O options you selected, test dongles are shipped already attached to the xPC TargetBox I/O connectors 1 through 6. The CAN dongle, for I/O option 308, is packed with the power cord.

Note, the bottom of the xPC TargetBox has a label that describes the configuration of this particular unit. This label indicates the I/O boards in your system, their base addresses, and the ports they are connected to. Replacement configuration labels are available in the lower compartment of the shipping case. Please contact your MathWorks representative if you need more configuration labels.

- 3** Attach the AC adapter and the AC power cord to the xPC TargetBox. If your xPC TargetBox includes I/O option 308, attach the CAN dongle to the CAN 1 and CAN 2 connectors.
- 4** Plug in the AC power cord to the wall outlet. When you do this, the internal self-test program for your xPC TargetBox starts running. The two user LEDs turn on, turn off, and then, if the self-test program is successful, turn back on.

See “Running the Self-Test Without a Monitor” on page 2-6.

- 5** After you have tried the self-test program, you can connect an SVGA video monitor to the xPC TargetBox. Run the self-test program again and observe the results using the scopes displayed on the monitor.

See “Running the Self-Test With a Monitor” on page 2-11.

- 6** Set up an xPC TargetBox for use with xPC Target.
- See “xPC Target Software Installation” on page 3-3.
  - See “xPC TargetBox Hardware Installation” on page 3-7.
  - See “Host PC to xPC TargetBox Communication” on page 3-14.

- 7** Create an xPC Target boot disk for BootFloppy mode.

See “Creating a Target Boot Disk” on page 3-22.

- 8** Boot your xPC TargetBox and test the connections to a host PC.

See “Booting an xPC TargetBox” on page 3-24.

## Features of xPC TargetBox

xPC TargetBox includes many features to help you prototype, test, and deploy real-time systems using xPC Target software. This section includes the following topics:

Hardware (p. 1-6)	Description of the hardware and external connections
Software (p. 1-8)	Description of the BIOS, DOS, FTP server, and a custom target application for system testing preinstalled on the flash disk
xPC TargetBox Boot Modes (p. 1-9)	Description of how to boot the kernel from the flash disk on the xPC TargetBox and download a target application from a host PC, or how to create a stand-alone target application and boot it from the flash disk on the xPC TargetBox
Communication (p. 1-10)	Description of how to communicate with an xPC TargetBox through standard peripherals, serial ports, and a network connection
I/O Options (p. 1-11)	Description of how to add to an xPC TargetBox the options that satisfy the needs of most xPC Target applications
Expected Background (p. 1-12)	Description of the MathWorks products you need to be familiar with before using an xPC TargetBox

### Hardware

The xPC TargetBox Version 1 hardware was designed to handle a wide range of applications and environments. It is a 100% PC-compatible, all-in-one embedded computer with the following characteristics:

**CPU performance** — Available systems include a Pentium MMX 266 MHz, Pentium III 400 MHz, or Pentium III 700 MHz processor. The performance of these systems is sufficient for 75% of all xPC Target applications found in office, laboratory, and mobile (field) environments.

**System memory** — A FlashRAM chip is mounted directly on the IDE connector, allowing you to boot the system without a floppy disk drive. This configuration requires no additional mechanical parts. However, you do need the external floppy disk drive or a network connection to transfer files to the system memory. This guide alternately refers to this system memory as flash disk and flash memory.

**PC/104 expansion bus** — I/O options are standard PC/104 boards.

**Small size** — The overall design makes xPC TargetBox ideal for use in mobile and field applications. Physical dimensions are the same for all units: 270 mm x 162 mm x 82 mm (6.8 in x 4 in x 2 in).

**Rugged system** — The xPC TargetBox has an aluminum enclosure that is anodized on the inside, is equipped with all necessary I/O connectors, and has the ability to operate under normal or harsh conditions without the need for cooling fans or other moving parts.

**Temperature range** — A low power design allows fanless operation with no mechanically rotating parts. Cooling holes in the enclosure are not necessary. The standard temperature range is 0°C to 60°C. For harsh conditions, an xPC TargetBox is available with an extended temperature range that can run from -40°C to 75°C (models 206 and 207) or from -40°C to 65°C (model 208).

**Internal power supply** — Onboard power supply with DC power input for voltages between 8 and 28 volts.

**External AC adapter** — The external AC power adapter supplied with the xPC TargetBox allows worldwide AC operation with an input of 110 to 230 volts, 50 Hz to 60 Hz.

An xPC TargetBox has capabilities that allow you to use it in a stand-alone mode of operation. Since the unit is powered by DC voltage, you can power it with a number of battery types that range from 8 to 28 volts DC.

**External floppy disk drive** — An external floppy disk drive is provided that you can connect to an xPC TargetBox using a standard 25-pin shielded cable (dedicated 25-pin port on the back panel). The external floppy disk drive does not need additional power because the xPC TargetBox provides power to the disk drive.

With this configuration, you can use the external floppy disk drive to

- On the xPC TargetBox, boot the xPC Target kernel from a boot floppy disk using the standard setup for xPC Target. When using the xPC TargetBox for mobile applications, you can load the kernel/application onto the flash disk, and then remove the external disk drive.
- Restore files to flash memory on the xPC TargetBox and recover from a system failure.

## Software

The xPC TargetBox is shipped with the following software on the flash disk:

**FreeDOS** — The xPC TargetBox uses FreeDOS to transfer stand-alone kernel/target application files to the flash disk, and to boot the kernel for downloading target applications from a host PC.

An xPC TargetBox does not include Microsoft Windows.

**FTP server** — If you have a target application that is larger than 1 MByte (xPC Target allows applications up to 16 MB) you cannot use the external floppy disk drive to transfer stand-alone applications to an xPC TargetBox. Additionally, if your application requires a rugged environment, using a mechanical drive is not suitable.

In these cases, you can connect an xPC TargetBox to a LAN or directly to a host PC with an Ethernet crossover cable, and use the FTP server on the flash disk to transfer files.

**Quick-booting the xPC TargetBox** — Normally the BIOS of a standard PC takes about 20 to 30 seconds to boot. With the QuickBoot capability, the BIOS boot time is reduced to about 2 seconds.

The QuickBoot capability provides a BIOS optimized for a particular system configuration. The optimized BIOS (QuickBoot binary) is preinstalled in the xPC TargetBox EEPROM.

---

**Note** The QuickBoot capability is available only with an xPC TargetBox using a Pentium III CPU (xPC TargetBox models 107, 108, 207, and 208).

---

**Target application for system testing** — The flash disk in the xPC TargetBox contains an xPC Target stand-alone application that self-tests the hardware and all I/O options installed in your xPC TargetBox. One of your first tasks is to run this self-test application and verify that the system is functioning correctly.

## **xPC TargetBox Boot Modes**

The xPC Target software includes the following boot mode for all target PC systems, including an xPC TargetBox.

- **BootFloppy mode** — Boot the xPC TargetBox from a boot floppy disk created by xPC Target on a host PC. When you boot the xPC TargetBox from this disk, the xPC Target kernel uses the resources on the target PC (CPU, RAM, and serial port or network adapter) without changing the files already stored on the flash disk.

xPC Target includes the following boot modes to use with an xPC TargetBox only. These additional boot modes are also available for general use with any target PC as part of the xPC Target Embedded Option.

- **DOSLoader mode** — Copy the xPC Target kernel to the flash disk using a floppy disk or the FTP server. After transferring the kernel and utility files, boot the kernel from the flash disk, and then download a target application from the host PC. The xPC TargetBox needs to be connected to a host computer with a serial or network cable.

- StandAlone mode — Copy the xPC Target kernel with a self-contained target application, using the FTP server. After transferring the files, you can reset the xPC TargetBox. The xPC TargetBox automatically boots the kernel and executes your target application without the need to communicate with a host PC.

To use the additional boot modes, see “Activating the Additional xPC TargetBox Boot Modes” on page 3-4.

## **xPC Target Embedded Option**

The xPC Target Embedded Option is an extension to xPC Target. You need to purchase a license for the xPC Target Embedded Option for the following:

- To use the DOSLoader and StandAlone boot modes on a different target PC than your xPC TargetBox or on multiple xPC TargetBoxes.
- To deploy stand-alone GUI applications on a host PC system that does not contain your licensed copy of xPC Target.

For more information about the xPC Target Embedded Option, see Embedded Option in the xPC Target User’s Guide.

## **Communication**

You can communicate with an xPC TargetBox through standard peripherals, serial ports, and a network connection.

Standard PC peripherals — There is support for standard peripherals accessible through standard connectors: SVGA interface (up to 1280 x 1024), PS/2 mouse, PS/2 keyboard, four RS-232 ports, and a standard parallel port.

With the addition of standard peripherals, you can use the xPC Target kernel command-line interface to enter commands for control, signal acquisition, and parameter tuning directly on an xPC TargetBox without the need for a host PC.

Network or serial connection to the host PC — Connect an xPC TargetBox to your host computer using either an RS-232 port (COM1 or COM2) or the onboard Ethernet controller (10BASE-T/100BASE-TX, Intel 82559ER).



You use the connection to the host PC to download a target application (BootFloppy and DOSLoader boot modes), or to transfer files (DOSLoader and StandAlone boot modes) to the flash disk using the FTP server.

## I/O Options

You can add up to three I/O options (boards) to an xPC TargetBox.

I/O expandability — The xPC TargetBox provides the ability to expand I/O connections through a standard PC/104 (ISA) and PC/104+ (PCI) expansion bus. The MathWorks currently offers seven I/O options you can purchase with your system.

If needed, you can add your own PC/104 I/O boards, but The MathWorks and MPL will not provide repair service for a system with your boards installed inside. If your system requires service, you must return it for repair in the same configuration in which it was shipped to you.

---

**Note** You can add up to two PC/104+ boards to your xPC TargetBox.

---

For more information about the available I/O options, see “xPC TargetBox I/O Options” on page 1-18.

I/O driver support — The xPC Target software contains driver blocks for all seven I/O options currently available for an xPC TargetBox. These options satisfy typical rapid prototyping requirements with A/D, D/A, digital I/O, CAN, counters, timers, encoders, and PWM. The drivers are represented by Simulink blocks in the xPC Target driver library. In addition, the library contains driver blocks for the user LEDs and the watchdog circuitry. Your interaction with these drivers is through Simulink blocks and their parameter dialog boxes.

You drag and drop blocks from the I/O library and connect I/O drivers to your Simulink model the same way as you would connect any standard Simulink block.

Internal connectors and cables — Shielded connectors and internal cables bring the PC/104 I/O board signals to the outside of an xPC TargetBox.

External connectors and cables — Shielded connectors are mounted onto the front and back panels. The connection layout is the same for all configurations. I/O options include shielded external cables with screw terminal boards.

Onboard CAN controller — The Intel 82527 CAN controller on the CPU board is not officially supported by the xPC TargetBox driver library. For applications that need CAN connections, add I/O option 308 to an xPC TargetBox.

## **Expected Background**

Before you use an xPC TargetBox, you should be familiar with

- Simulink and Stateflow — Using Simulink and Stateflow to create dynamic models as block diagrams, and simulating those models in Simulink.
- Real-Time Workshop — 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 another programming language to create real-time applications.
- xPC Target — Communication between a host and target PC. Using the MATLAB® command-line interface or xPC Target graphical interfaces to tune parameters and acquire signals.

If you are a new xPC TargetBox user — Begin reading and testing your system using the procedures in Chapter 2, “Initial Loopback Test” and then complete the installation following procedures in Chapter 3, “Regular Use”.

Continue by reading this chapter and the Introduction in the Getting Started with xPC Target documentation. These chapters will give you an overview of the xPC TargetBox features and the xPC Target development environment.

If you are an experienced xPC TargetBox user — The xPC TargetBox User’s Guide gives detailed information about the I/O options available with an xPC TargetBox. This guide also explains how to use the FTP server to transfer files from a host PC, and how to create and run stand-alone applications.

## System Requirements

The host PC is your desktop or notebook computer where you install MATLAB, Simulink, Stateflow and Stateflow Coder (optional), Real-Time Workshop, xPC Target, and xPC Target Embedded Option (optional). This section includes the following topics:

Software Requirements for a Host PC (p. 1-14)	Description of the software requirements, including MATLAB, Simulink, Real-Time Workshop, xPC Target, and the C/C++ compiler
Hardware Requirements for a Host PC (p. 1-15)	Description of the hardware requirements, including desktop or notebook
xPC TargetBox Software (p. 1-15)	Description of the software, including FreeDOS, QuickBoot capability for Pentium III based systems, and a custom target application for testing your system
xPC TargetBox Hardware (p. 1-16)	Description of the hardware, including Pentium CPU, flash disk, standard PC connections, slots for three I/O boards, and an external floppy disk drive
xPC TargetBox I/O Options (p. 1-18)	Description of the I/O options, including analog input, analog output, digital I/O, encoders, counter/timers, and interrupt inputs

## Software Requirements for a Host PC

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

Software	Description
Operating system	A Microsoft Windows platform supported by The MathWorks
MATLAB	Version 7.3 (Release 2006b)
Simulink	Version 6.5 (Release 2006b). Create physical models with block diagrams.
Stateflow (optional)	Version 6.5 (Release 2006b). Add state charts to a Simulink model.
Real-Time Workshop	Version 6.5 (Release 2006b). Generate code from Simulink models.
Stateflow Coder (optional)	Version 6.5 (Release 2006b). Use with Real-Time Workshop to generate code from Simulink models with state charts.
C programming language compiler	Microsoft Visual C/C++ Version 6.0, 7.1, or 8.0, or Open Watcom C/C++ Version 1.3. For optimal performance, use the professional versions of the Microsoft Visual tools in your environment. Use with Real-Time Workshop and xPC Target to generate executable code to run with the xPC Target kernel.
xPC Target	Version 3.1 (Release 2006b) with the most recent xPC Target library update. Each xPC TargetBox requires an active software license for xPC Target.
xPC Target Embedded Option (optional)	Version 3.1 (Release 2006b). Create graphical interfaces to a target application using an API or COM API interface. The xPC Target Embedded Option includes a license for unlimited deployment of applications on multiple target PCs.

## Hardware Requirements for a 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. The network can be as simple as a crossover Ethernet cable from the host PC to an xPC TargetBox.
CPU	Intel Pentium, AMD Athlon, or later
Peripherals	Hard disk drive with 50 MB of free space: 23 MB for xPC Target, 20 MB for xPC Target Embedded Option, and 7 MB for HTML documentation  3.5-inch floppy disk drive for creating a disk to boot the xPC TargetBox and transferring files to the flash disk  CD-ROM drive for installing MathWorks software
RAM	128 MB or more

## xPC TargetBox Software

The following table lists the software included with an xPC TargetBox.

Software	Description
BIOS	PC compatible. Standard BIOS with Pentium MMX systems (models 106 and 206). QuickBoot BIOS with Pentium III systems (models 107, 108, 207, 208).
Operating system	FreeDOS preinstalled on the flash disk.

<b>Software</b>	<b>Description</b>
File transfer	FTP server preinstalled on the flash disk. Copy stand-alone target applications from a host PC to an xPC TargetBox using a network connection. You do not need a floppy disk drive.
Self-test target application	Customized target application for testing the specific I/O options installed in your xPC TargetBox.

## **xPC TargetBox Hardware**

xPC TargetBox is designed and manufactured by MPL exclusively for The MathWorks. For a detailed hardware specification, refer to the xPC TargetBox hardware reference documentation provided in the shipping case from MPL. The following table summarizes the hardware.

<b>Hardware</b>	<b>Description</b>
Communication	Serial ports (SERIAL 1, SERIAL 2) with a 9-pin D-sub connector, and network connection with an Ethernet RJ-45 connector
CPU	Pentium MMX systems (models 106 and 206) or Pentium III systems (models 107, 108, 207, 208)
Peripherals	You need to supply an SVGA monitor, PS/2 keyboard, and PS/2 mouse.
RAM	128 MB
Nonvolatile memory	32 MB FlashRAM module (IDE flash disk)
I/O expansion slots	Room for three PC/104 expansion boards. The xPC TargetBox is not expandable beyond adding I/O boards.

Hardware	Description
External I/O connectors	<p>Front and back panels with shielded I/O connectors for I/O board options</p> <p>Onboard CAN bus connector (not currently supported). Use IO option 308 for a CAN bus interface.</p> <p>Connectors for standard PC peripherals (monitor, mouse, keyboard, four RS-232 ports, one parallel port, and RJ-45 Ethernet)</p>
External power source	External AC adaptor with an output voltage of +19 volts DC for worldwide operation. The type of power cord provided with an xPC TargetBox depends on the location to which the unit is shipped.
External floppy disk drive	One external 3.5-inch floppy disk drive with a dedicated parallel port connector

The following table lists available xPC TargetBox base systems.

Standard Unit Name	Description
xPC TargetBox 106	266 MHz Pentium MMX, 128 MB RAM, 32 MB FlashRAM. Standard temperature from 0°C to 60°C.
xPC TargetBox 107	400 MHz Pentium III, 128 MB RAM, 32 MB FlashRAM with QuickBoot capability. Standard temperature from 0°C to 60°C.
xPC TargetBox 108	700 MHz Pentium III, 128 MB RAM, 32 MB FlashRAM with QuickBoot capability. Standard temperature from 0°C to 60°C.

The following table lists available xPC TargetBox base systems with extended temperature ranges.

<b>Extended Temperature Unit Name</b>	<b>Description</b>
xPC TargetBox 206	266 MHz Pentium MMX, 128 MB RAM, 32 MB FlashRAM. Extended temperature from -40°C to 75°C.
xPC TargetBox 207	400 MHz Pentium III, 128 MB RAM, 32 MB FlashRAM with QuickBoot capability. Extended temperature from -40°C to 75°C.
xPC TargetBox 208	700 MHz Pentium III, 128 MB RAM, 32 MB FlashRAM with QuickBoot capability. Extended temperature from -40°C to 65°C.

### **xPC TargetBox I/O Options**

The following table lists the available I/O options for an xPC TargetBox. You can add up to three I/O options in a single xPC TargetBox.

Each I/O option includes an installed PC/104 board, one or two internal cables connected to either the front or back panels, one or two external I/O cables, one or two screw terminal boards (except for the CAN I/O option), one or two test dongles, and I/O board reference documentation.

The label on the bottom of the xPC TargetBox indicates the I/O boards in your system, their base addresses, and the ports to which they are connected.



I/O Option Name	Description
<p>xPC TargetBox IO 301 Diamond-MM-32-AT or Diamond-MM-32X-AT</p> <p>Extended temperature -40°C to 85°C</p>	<p>Analog input (A/D) — 32 single-ended, 16 differential, or 16 single-ended/8 differential (16-bit) channels with a maximum sample rate of 200 kHz (5 <math>\mu</math>s). Channel coupling selected with jumpers on board.</p> <p>If you do not specify a channel coupling when purchasing an xPC TargetBox, the coupling is set to 16 differential channels.</p> <p>Unipolar and bipolar input ranges of 0-10 V, 0-5 V, 0-2.5 V, 0-1.25 V, 0-0.625 V, <math>\pm 10</math> V, <math>\pm 5</math> V, <math>\pm 2.5</math> V, <math>\pm 1.25</math> V, and <math>\pm 0.625</math> V selected by software.</p> <p>Analog output (D/A) — 4 (12-bit) channels with a 6 <math>\mu</math>s settling time. Output range set to 0-10 V, 0-5 V, <math>\pm 10</math> V, or <math>\pm 5</math> V with jumpers on the board.</p> <p>If you do not specify an output range when purchasing an xPC TargetBox, the range is set to <math>\pm 10</math> V.</p> <p>Digital I/O — 24 digital I/O lines divided into three groups with 8 bits each.</p> <p>xPC TargetBox does not support the counters on this board.</p>
<p>xPC TargetBox IO 302 Diamond Ruby-MM-1612</p> <p>Extended temperature -40°C to 85°C</p>	<p>Analog output (D/A) — 16 (12-bit) channels with a 6 <math>\mu</math>s settling time. Output range for each group of 8 channels set to 0-10 V, 0-5 V, 0-2.5 V, <math>\pm 10</math> V, <math>\pm 5</math> V, or <math>\pm 2.5</math> V with jumpers on the board.</p> <p>If you do not specify a range when purchasing an xPC TargetBox, the range is set to <math>\pm 10</math> V.</p> <p>Digital I/O — 24 TTL digital I/O lines divided into three groups with 8 bits each</p> <p>xPC TargetBox does not support the external trigger on this board.</p>

<b>I/O Option Name</b>	<b>Description</b>
<p>xPC TargetBox IO 303 Diamond Ruby-MM-416</p> <p>Standard temperature 0°C to 70°C</p>	<p>Analog output (D/A) — 4 (16-bit) channels with a 10 <math>\mu</math>s settling time. Output range for each channel set independently to 0-10 V, <math>\pm</math>10 V, or <math>\pm</math>5 V with jumpers on the board.</p> <p>If you do not specify a range when purchasing an xPC TargetBox, the range is set to <math>\pm</math>10 V.</p> <p>Digital I/O — 24 TTL digital I/O lines divided into three groups with 8 bits each.</p> <p>xPC TargetBox does not support the external trigger on this board.</p>
<p>xPC TargetBox IO 304 Diamond Onyx-MM</p> <p>Extended temperature -40°C to 85°C</p>	<p>Digital I/O — 48 TTL digital I/O lines divided into six groups with 8 bits each. Each group set to either input or output.</p> <p>xPC TargetBox does not support the counter/timers and external interrupts on this board.</p>
<p>xPC TargetBox IO 305 Diamond Quartz-MM-10</p> <p>Standard temperature 0°C to 60°C</p>	<p>Counters — 10 (16-bit) general purpose counters for pulse-train generation and pulse-width measurements. Maximum clock frequency 4 MHz.</p> <p>Digital I/O — 8 TTL digital input lines and 8 TTL digital output lines.</p> <p>Interrupt — 1 digital interrupt line.</p> <p>Note that this option is no longer available. Consider the IO 309 option as an alternative.</p>

<b>I/O Option Name</b>	<b>Description</b>
<p>xPC TargetBox IO 306 Real Time Devices DM6814</p> <p>Standard temperature 0°C to 70°C</p>	<p>Encoders — 3 up/down (16-bit) counters with a maximum input rate of 1 MHz.</p> <p>Digital I/O — 18 TTL digital input-only lines and 6 TTL input or output lines.</p> <p>Interrupts — 2 digital interrupt lines.</p> <p>xPC TargetBox does not support the counter/timers on this board.</p>
<p>xPC TargetBox IO 308 Softing CAN-AC2-104</p> <p>Extended temperature -40°C to 85°C</p>	<p>CAN field bus — Two CAN channels. SJA 1000 controller for CAN 2.0A (11-bit standard frames) and CAN 2.0B (29-bit extended frames).</p> <p>Both CAN channels are set to be nonterminated. If you locate one or both of the CAN channels at the end of a CAN bus, you must provide termination by placing the correct resistor directly at the connector pins.</p>
<p>xPC TargetBox IO 309 MPL PATI (PowerPC controlled Analog and Timing I/O Intelligence)</p> <p>Standard temperature 0°C to 70°C</p> <p>Extended temperature -40°C to 85°C (optional)</p>	<p>xPC TargetBox supports the 32 TPU channels of this board for</p> <ul style="list-style-type: none"> <li>• PWM measurement</li> <li>• PWM generation</li> <li>• Time base selection</li> <li>• Incremental encoder measurement</li> <li>• Digital I/O</li> </ul>



# Initial Loopback Test

---

This chapter describes how to self-test the xPC TargetBox. xPC TargetBox ships with a custom target application to self-test the I/O options installed. This self-test uses dongles attached to the connectors to loopback the output signals to inputs. This chapter also describes how to connect hardware to the xPC TargetBox. This chapter includes the following sections:

Installation for Loopback Testing  
(p. 2-2)

Describes how to connect an AC power adapter and a monitor to an xPC TargetBox

Connecting Hardware to an xPC TargetBox (p. 2-18)

Describes how to use screw terminal boards to interface your hardware with the xPC TargetBox I/O options

## Installation for Loopback Testing

When you receive your xPC TargetBox, your first task is to run a target application to self-test your particular xPC TargetBox with your selected I/O options. This section includes the following topics:

Unpacking the Shipping Case (p. 2-2)	How to take out the xPC TargetBox from the plastic shipping case
Running the Self-Test Without a Monitor (p. 2-6)	How to connect the AC power adaptor and run the self-test
Running the Self-Test With a Monitor (p. 2-11)	How to connect a monitor to the xPC TargetBox and use the self-test to isolate problems
Troubleshooting the Loopback Self-Test (p. 2-15)	How to interpret the information from the self-test to solve problems identified with this test

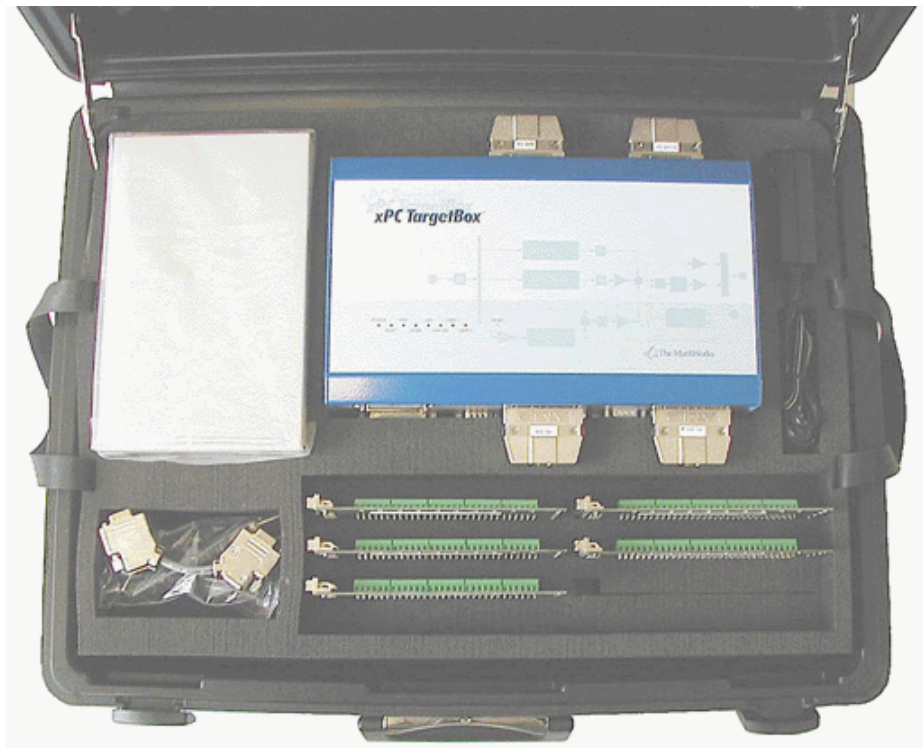
### Unpacking the Shipping Case

The xPC TargetBox is delivered in a plastic shipping case. This shipping case includes everything you need to power up an xPC TargetBox and run the self-test.

- 1** Take out the xPC TargetBox shipping case from the shipping case. Check visually for any physical damage.
- 2** Place the shipping case on a flat surface so you can read the labels on either side of the shipping case handle.



- 3** Press the bottom of both latches and open the top lid.



**Contents of the top layer** — xPC TargetBox with attached test dongles, Read First document, external floppy disk drive, external AC power adaptor, AC power cord, and screw terminal boards.

- 4** Remove the external floppy disk drive to uncover the drive cable.





**Contents of floppy drive compartment** — External floppy disk drive cable.

- 5 Lift up the upper layer of the shipping case by its handles and place the upper layer aside.

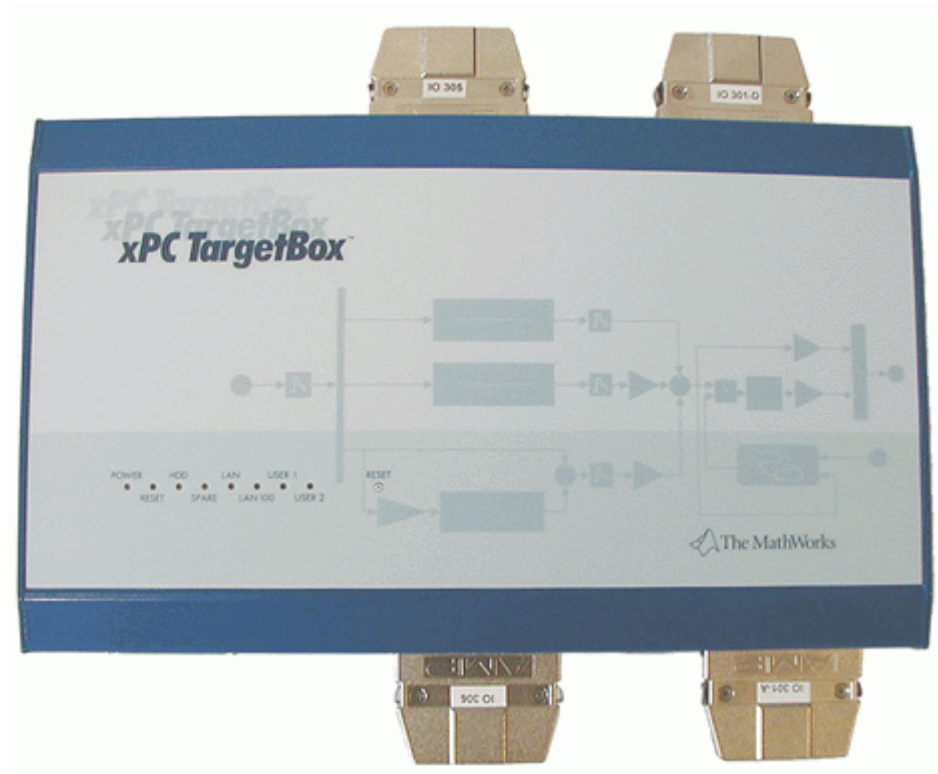


- 6 **Contents of the lower layer** — External I/O cables, xPC TargetBox User's Guide, MPL hardware reference manual, and I/O reference manuals.
- 7 If any parts seem to be missing, contact your MathWorks representative.

### **Running the Self-Test Without a Monitor**

A self-test is preinstalled on the xPC TargetBox flash disk. This test is a custom xPC Target application to exercise the I/O options in your system. Run this application to test your system initially. You can also run this application any time you are not sure about the performance of your xPC TargetBox.

- 1 From the top layer of the xPC TargetBox shipping case, take out the xPC TargetBox. Check visually for any physical damage.



- 2 Turn the xPC TargetBox over, and on the bottom, find the configuration label. (There is no reason to worry, the xPC TargetBox is a rugged product intended to be handled thusly.) Check for the correct xPC TargetBox model and I/O options that you ordered. Also note the industrial standards for which your xPC TargetBox is certified.



Replacement configuration labels are available in the lower compartment of the shipping case.

- 3 Turn the xPC TargetBox back over and place on a desk with the top label up. Make sure the test dongles are tightly plugged in.
- 4 If your xPC TargetBox includes the I/O option 308 (CAN-AC2-104), remove the CAN dongle from the upper compartment of the xPC TargetBox shipping case and attach it to the CAN 1 and CAN 2 connectors of your xPC TargetBox.
- 5 From the top layer of the case, remove the external AC power adapter. Check visually for any physical damage.



- 6 Place the AC power adapter near the xPC TargetBox. Connect the loose end, with the green connector block, to the power input on the xPC TargetBox.



- 7 From the top compartment, remove the AC power cord.

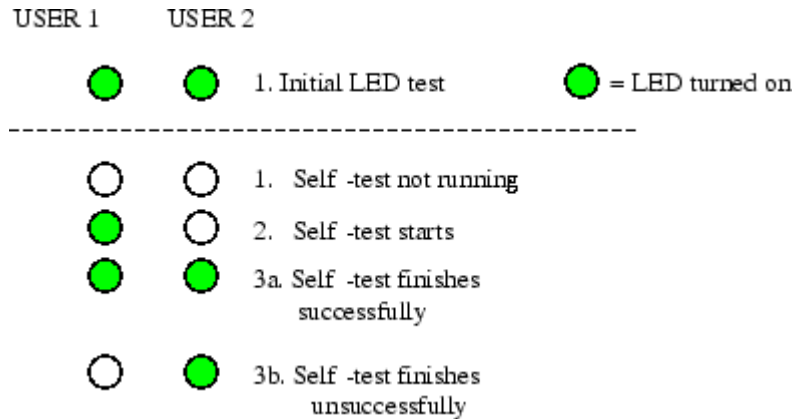


- 8 Plug the socket end of the cable into the external AC power adapter.



You are now ready to start the xPC TargetBox for self-testing. Because the xPC TargetBox does not have a power switch, the system starts to operate as soon as you plug the AC power cord into a wall outlet.

- 9 Plug the power cord into the wall outlet and observe the LEDs on the top of the xPC TargetBox. The LEDs light with the following sequence:
  - Power on — Red Reset LED turns on momentarily, and then the yellow Power LED turns on and stays on. If you have a network connection, the green LAN and LAN100 LEDs turn on. The green HDD LED turns on when the BIOS accesses the flash disk.
  - DOS test — Both USER 1 and USER 2 LEDs turn on.
  - xPC Target self-test — Kernel loads (both USER LEDs off), self-test starts (USER 1 on), self-test is successful (both USER LEDs on), or self-test is unsuccessful (USER 2 on).



**10** Observe the sequence of LEDs turning on and off.

- If none of the LEDs turns on, immediately unplug the power cord. Check for loose connectors (AC power adapter to xPC TargetBox connector, AC adapter power cord).

Plug in the power cable again. If the system does not boot up as described above, unplug the power cable from the wall outlet and contact a MathWorks representative for help.

- If the self-test fails (USER 1 LED does not turn on), try to isolate the problem using a monitor connected to the xPC TargetBox. See “Running the Self-Test With a Monitor” on page 2-11.

Note that if your xPC TargetBox includes the I/O option 308 (CAN-AC2-104), you must ensure that it is attached to the xPC TargetBox. See step 4 of “Running the Self-Test Without a Monitor” on page 2-6.

## Running the Self-Test With a Monitor

A self-test is preinstalled on the xPC TargetBox flash disk. This test is a custom xPC Target application to exercise the I/O options in your system. Run this application to test your system initially. You can later run this application any time you are not sure about the performance of your xPC TargetBox.

After you run the self-test without a monitor, you can try running the self-test with a monitor:



- 1 Set up the xPC TargetBox as described in “Running the Self-Test Without a Monitor” on page 2-6. The AC power cord should be unplugged.

Find a monitor with a resolution of at least 640-by-480 pixels, 16 colors, and an SVGA 15-pin (3 rows) connector. Connect the VGA cable of the monitor to the VGA connector on the back panel of the xPC TargetBox. Plug in the monitor power cable and turn on the monitor. Plug in the monitor power cable and turn on the monitor.



You are now ready to start the xPC TargetBox for self-testing. Because the xPC TargetBox does not have a power switch, the system starts to operate as soon as you plug the AC power cord into a wall outlet.

- 2 Plug in the AC power cord and observe the LEDs on the top of the xPC TargetBox and the monitor screen.

The sequence of LEDs turning on and off should follow the sequence described in “Running the Self-Test Without a Monitor” on page 2-6. You should see a display on the monitor for the first time. The initial content you see on the screen depends on the model of your xPC TargetBox.



**xPC TargetBox 106/206** — The monitor shows the typical system (BIOS) boot messages (Detecting RAM, finding media devices (floppy, flash), and displaying found PCI devices). This takes about 10 seconds. After this, DOS is loaded from the internal flash disk. The stand-alone xPC Target self-test application, which has been preloaded on the flash disk, begins.

**xPC TargetBox 107/207/108/208** — Because these boxes are equipped with a QuickBoot BIOS that boots in a very short time, any output to the monitor (VGA) is suppressed. The first message that you can see on the monitor is when FreeDOS is loaded. After FreeDOS loads from the internal flash disk, the stand-alone xPC Target self-test application, which has been loaded on the flash disk, begins.

**3** Observe the monitor screen.

The monitor should show a simple graphical interface for the self-test. Confirm that the self-test is running by observing in the upper left window that the running time is increasing. A running self-test is the first major sign that the xPC TargetBox is operating properly.

<code>Loaded App: xpctbtst Memory: 124MB Mode: RT, multi Logging: t StopTime: 5 d SampleTime: 0.001 AverageTET: None Execution: stopped</code>	<code>Scope: acquisition of scope 4 is running Scope: acquisition of scope 4 is running System: execution started (sample time: 0.001000) System: execution stopped at 4.000000 Scope: 5, set to state 'Interrupted' Scope: 2, set to state 'Interrupted' Scope: 3, set to state 'Interrupted' Scope: 1, set to state 'Interrupted' Scope: 4, set to state 'Interrupted'</code>
--	---

<code>F1 SC5 I0308; Port: 0x240; Result: 0</code>	<code>F2 SC2 I0301 A/D; Port: 0x300; Result: 1</code>
<code>F3 SC3 I0301 DIO; Port: 0x300; Result: 1</code>	<code>F4 SC1 I0305; Port: 0x340; Result: 1</code>
<code>F5 SC4 I0306; Port: 0x280; Result: 1</code>	

#### 4 Observe the lower section of the monitor screen.

The monitor screen displays a window for each connector attached to an I/O option. The first value is the I/O option number and connector (where applicable), the second is the base address (this is the base address for the board for which the test application was created). For example, I0301 A/D; Port: 0x300 is the analog connector for the IO 301 option and I0301 DIO; Port 0x300 is the digital I/O for the same I/O option. This information should match the configuration label on the bottom of your

xPC TargetBox. The **Result** label has a numerical value that indicates the status of the loopback test.

- A numerical value of 1 indicates that the loopback self-test was successful for that particular I/O option. If all numerical values are 1, the self-test is running successfully and your xPC TargetBox is fully functional. Unplug the power cord and proceed to Chapter 3, “Regular Use”
- If one of the numerical values is 0, the loopback test detected a failure when exercising a particular I/O option. Unplug the power cord. Check that, if your xPC TargetBox includes the I/O option 308 (CAN-AC2-104), this option is attached to the xPC TargetBox. See step 4 of “Running the Self-Test Without a Monitor” on page 2-6. Rerun the test.

If the loopback test still detects a failure, proceed to the following topic, “Troubleshooting the Loopback Self-Test” on page 2-15.

## Rebooting or Restarting the xPC TargetBox

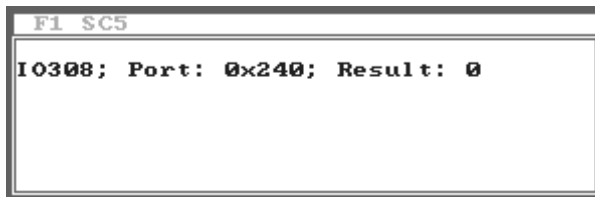
You can reboot or restart the xPC TargetBox in two ways:

- AC power cord and adapter — The xPC TargetBox does not have an on/off switch. To power the xPC TargetBox on and off, plug and unplug the AC power cord.
- Reset button — As long as the xPC TargetBox is powered on, you can reboot it by pressing the RESET button on top.

## Troubleshooting the Loopback Self-Test

The self-test exercises the I/O options in your xPC TargetBox and identifies problems. With the information from the self-test you can locate where a problem occurred.

- 1 On the target screen, find the window or windows where a value of 0 is shown. For example, the following screen shows that I/O option 308 failed (Result: 0) the test.



- 2** Write down the I/O options that show a numerical value of 0.
- 3** Unplug the AC power cord.
- 4** Using the following table and the I/O options you wrote down, determine the corresponding board names.

<b>IO Option</b>	<b>Board Name</b>	<b>Manufacturer</b>
IO 301	Diamond-MM-32-AT or Diamond-MM-32X-AT	Diamond
IO 302	Ruby-MM-1612	Diamond
IO 303	Ruby-MM-416	Diamond
IO 304	Onyx-MM	Diamond
IO 305	Quartz-MM-10	Diamond
IO 306	DM6814	Real Time Devices
IO 308	CAN-AC2-104	Softing
IO 309	MPL PATI	MPL

- 5** Turn the xPC TargetBox over and find the configuration label on the bottom. Look for a listing of I/O options for this xPC TargetBox. The I/O options are attached to the connectors labeled I/O 1 through I/O 6, CAN1, and CAN2.

Again, using the I/O option numbers, locate the connector number on the configuration label, and then locate the corresponding I/O connector on the front or back panel of the xPC TargetBox. The connectors on the front and back panels are marked I/O 1 through I/O 6 and CAN1 through CAN3.

- 6 Press the two latches on the test dongle connected to the corresponding I/O connector, remove it from the connector, and then reconnect it.



Removing and reattaching a test dongle ensures that the dongle is not loose.

- 7 Repeat the above step for each test dongle with a corresponding self-test that failed.
- 8 Power up the system again and let the system self-test run another time. If some numerical values in the target screen still show values of 0, unplug the system and contact Technical Support at The MathWorks for help.

## Connecting Hardware to an xPC TargetBox

Every practical xPC Target and xPC TargetBox application has physical connections to the hardware you are testing (plant, for rapid control prototyping, or controller, for hardware-in-the-loop simulation). These connections are between the xPC TargetBox I/O options and the external hardware. This section describes how to make these connections:

Introduction to I/O Options (p. 2-18)	How to determine the electrical and mechanical limitations of your hardware and xPC TargetBox I/O options before making any connections
Configuration Label for I/O Options (p. 2-19)	How to identify all I/O options and associated I/O connectors
Planning I/O Hardware Connections (p. 2-27)	How to read hardware manuals and draw a diagram of the connections
Connecting Hardware to Screw Terminal Boards (p. 2-28)	How to attach cables between an xPC TargetBox and screw terminal boards

### Introduction to I/O Options

The label on the bottom of your xPC TargetBox indicates the I/O options in your system and the connector ports to which these options are connected. An xPC TargetBox is shipped with external cables and screw terminal boards. You use these terminal boards to connect your equipment to the I/O boards in your system.

---

**Caution** You must be extremely careful that you do not damage an I/O option or even the entire xPC TargetBox. You can damage an I/O option by making a wrong connection or by not connecting hardware according to the specifications of the I/O option. Because the physical connections depend on your application, you must make sure on your own that the electrical and mechanical connections are within the boundaries of the I/O options. See the I/O board user's manual from the original manufacturer for each particular I/O option.

---

**External connectors for I/O options** — Each I/O option installed in your xPC TargetBox is accessible through one or two external I/O connectors on the front and back panels of the xPC TargetBox. These connectors are clearly marked I/O 1 through I/O 6 and CAN1 through CAN3. The first six connectors are of type SCSI-II with 50 pins while the three CAN connectors have 9 pins. The layout of the 50 pins depends on which I/O option is internally connected to which I/O connector on the outside of the xPC TargetBox.

For information about the pin layout for a specific I/O option, see Chapter 5, “xPC TargetBox I/O Options Reference”.

## **Configuration Label for I/O Options**

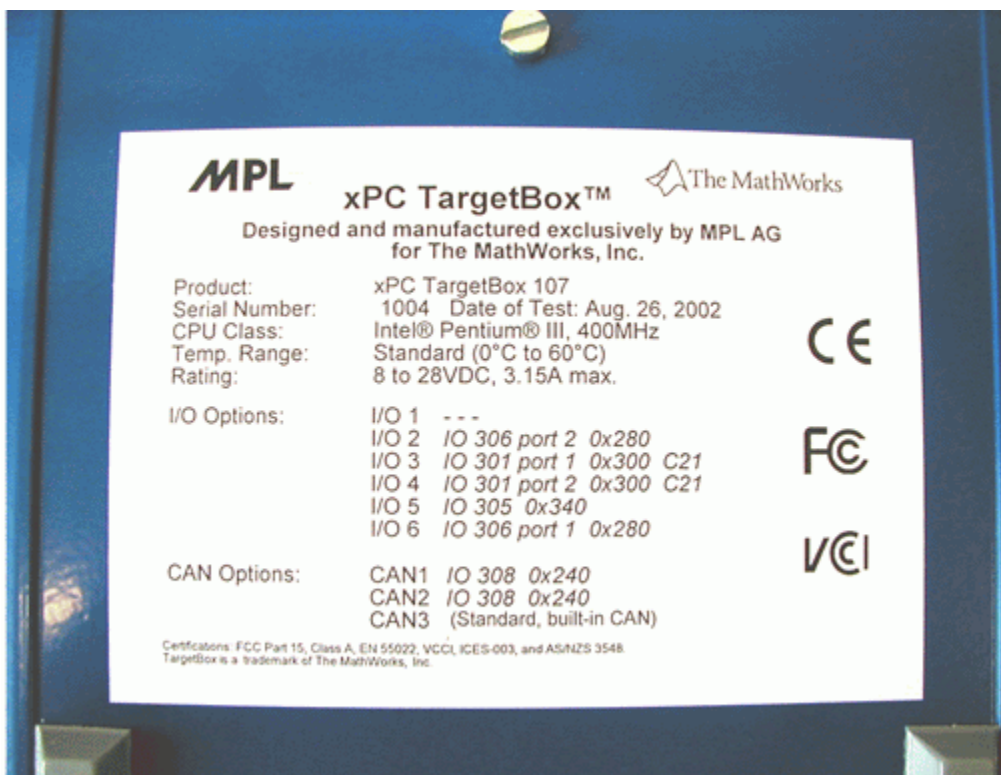
The layout of the I/O connectors on the xPC TargetBox is the same for all models and is independent of the I/O options installed in your xPC TargetBox. The mapping of which I/O option is connected to which I/O connector is listed on the xPC TargetBox configuration label on the bottom of the xPC TargetBox.

---

**Caution** Always unplug the xPC TargetBox when locating the label.

---

Three additional copies of these labels are in the Read First envelope found in the upper compartment of the shipping case. These additional labels are for your reference. You can either use them for your documentation materials or attach them to your application setup at a convenient location.



**Product** — Product name (xPC TargetBox) and model number (106, 107, 108, 206, 207, 208)

**Serial Number** — Serial number with date of system test

**CPU class** — CPU name (Pentium MMX, Pentium III) and speed (266 MHz, 400 MHz, 700 MHz)



**Temp. Range** — Standard (0°C to 60°C), Extended models 206 and 207 (-40°C to 75°C), or Extended model 208 (-40°C to 65°C)

**Rating** — Always 8 to 28 VDC, 3.15 A max

**I/O Options** — Each line on the label corresponds to a connector on the xPC TargetBox (I/O 1 through I/O 6) with the following fields:

- The I/O option associated with the connector (IO 301, IO 302, IO 303, IO 304, IO 305, IO 306).
- For options with more than one connector, a distinction between connectors (port 1, port 2).
- The base address of the I/O option (0x220, 0x240, 0x280, 0x300).
- If an I/O option (I/O board) includes jumper settings, the label includes a unique code to identify in which position the hardware jumpers are set. See the table, .

**CAN Options** — Each line on the label corresponds to a CAN connector on the xPC TargetBox (CAN1 through CAN3):

- The I/O option associated with the connector (IO 308)
- The base address of the I/O option (0x220, 0x240, 0x280, 0x300)

The following table lists all I/O options with the number of associated I/O connectors and possible jumper codes or switch settings. For a description of the I/O options, see “xPC TargetBox I/O Options” on page 1-18.

**Jumper Codes or Switch Settings for I/O Options**

<b>I/O Option</b>	<b>Number of Connectors</b>	<b>Jumper Codes or Switch Settings</b>
IO 301 Diamond-MM-32-AT or Diamond-MM-32X-AT	2	<p><b>Analog input (A/D) jumper codes</b> (first number after the C)</p> <ul style="list-style-type: none"> <li>• C1# — 32 single-ended</li> <li>• C2# — 16 differential</li> <li>• C3# — 16 single-ended/8 differential</li> </ul> <p><b>Analog output (D/A) jumper codes</b> (second number after the C indicates the output voltage range for the 4 D/A channels)</p> <ul style="list-style-type: none"> <li>• C#1 — ±10 V</li> <li>• C#2 — ±5 V</li> <li>• C#3 — 0-10 V</li> <li>• C#4 — 0-5 V</li> </ul> <p>The default configuration is C21, selecting 16 differential input channels and ±10 V for the output range on the 4 output channels.</p>

**Jumper Codes or Switch Settings for I/O Options (Continued)**

<b>I/O Option</b>	<b>Number of Connectors</b>	<b>Jumper Codes or Switch Settings</b>
IO 302 Ruby-MM-1612	1	<p data-bbox="813 401 1329 529"><b>Analog output (D/A) jumper codes</b> (first number after the C indicates the output voltage range for the first 8 D/A channels)</p> <ul data-bbox="813 564 1329 833" style="list-style-type: none"> <li>• C1# — <math>\pm 10</math> V</li> <li>• C2# — <math>\pm 5</math> V</li> <li>• C3# — <math>\pm 2.5</math> V</li> <li>• C4# — 0-10 V</li> <li>• C5# — 0-5 V</li> <li>• C6# — 0-2.5 V</li> </ul> <p data-bbox="813 868 1329 963">The third character (second number) in the code indicates the output voltage range for the second 8 channels.</p> <p data-bbox="813 980 1329 1076">The default configuration is C11, selecting <math>\pm 10</math> V for the output range on the 16 output channels.</p> <p data-bbox="813 1093 1329 1246">The following lists the allowed output voltage range combinations for the two channel banks. <b>B1</b> is the first bank of channels, <b>B2</b> is the second bank of channels.</p>

**Jumper Codes or Switch Settings for I/O Options (Continued)**

I/O Option	Number of Connectors	Jumper Codes or Switch Settings		
		B1	B2	Configuration
		±10 V	±10 V	C11
		0-10 V	±10 V	C41
		±5 V	±5 V	C22
		±2.5	±5 V	C32
		0-5 V	±5 V	C52
		0-2.5 V	±5 V	C62
		±5 V	±2.5 V	C23
		±2.5 V	±2.5 V	C33
		0-5 V	±2.5 V	C53
		0-2.5 V	±2.5 V	C63
		±10 V	0-10 V	C14
		0-10 V	0-10 V	C44
		±5 V	0-5 V	C25
		±2.5 V	0-5 V	C35
		0-5 V	0-5 V	C55
		0-2.5 V	0-5 V	C65
		±5 V	0-2.5 V	C26
		±2.5 V	0-2.5 V	C36
		0-5 V	0-2.5 V	C56
		0-2.5 V	0-2.5 V	C66

**Jumper Codes or Switch Settings for I/O Options (Continued)**

<b>I/O Option</b>	<b>Number of Connectors</b>	<b>Jumper Codes or Switch Settings</b>
IO 303 Ruby-MM-416	1	<p><b>Analog output (D/A) jumper codes</b> (first number after the C indicates the output voltage range for the first D/A channel)</p> <ul style="list-style-type: none"> <li>• C1### — ±10 V</li> <li>• C2### — ±5 V</li> <li>• C3### — 0-10 V</li> </ul> <p>The third, fourth, and fifth characters in the code indicate the output voltage range for the second, third, and fourth D/A channels.</p> <p>The default configuration is C1111, selecting ±10 V for the output range on the four output channels.</p>
IO 304 Onyx-MM	2	None
IO 305 Quartz-MM-10	1	None
IO 306 DM6814	2	None

**Jumper Codes or Switch Settings for I/O Options (Continued)**

<b>I/O Option</b>	<b>Number of Connectors</b>	<b>Jumper Codes or Switch Settings</b>
IO 308 CAN	Special connectors	None
IO 309 MPL	1	<p><b>Slot selection configuration</b> (first number after the S indicates the switch number (1), second number indicates the switch)</p> <ul style="list-style-type: none"> <li>• S1-1 and S1-2 — Selects the stacking position in which the board is found.               <ul style="list-style-type: none"> <li>With one board in the system                   <ul style="list-style-type: none"> <li>▪ S1-1 — off, S1-2 — off</li> </ul> </li> <li>With two boards in the system                   <ul style="list-style-type: none"> <li>▪ First board: S1-1 — off, S1-2 — off</li> <li>▪ Second board: S1-1 — on, S1-2 — off</li> </ul> </li> </ul> </li> <li>• S3 — off</li> <li>• S4 — on (enable reboot)</li> <li>• S5 — off</li> <li>• S6 — off</li> <li>• S7 — off</li> <li>• S8 — off</li> </ul> <p>See the MPL PATI user documentation for a description of the DIP switch S1. You can add up to two PC/104+ boards to your xPC TargetBox.</p>

When you have identified all I/O options and associated I/O connectors on your xPC TargetBox, you are ready to connect the xPC TargetBox through its I/O connectors to your hardware under test. See “Planning I/O Hardware Connections” on page 2-27.

## Planning I/O Hardware Connections

To ensure correct operation, the shipping case includes an I/O board user's manual from the original manufacturer for each particular I/O option. These manuals are located in the lower compartment of the xPC Target Box shipping case. Be sure to consult these manuals to verify that your I/O connections comply with the manufacturer's specifications:

- 1 Make a drawing of the I/O connections you want to establish.
- 2 Verify that the voltage levels, current loads, and so forth are in accordance with the specification of a particular I/O option.

---

**Note** Neither The MathWorks nor MPL will be liable for any damage resulting from connecting hardware to an xPC TargetBox in any way that is not in compliance with the I/O board specification from the original manufacturer.

---

- 3 From the following list determine the hardware manuals for your particular I/O options:

IO 301	Diamond Systems DIAMOND-MM-32-AT User Manual V2.61 or Diamond Systems DIAMOND-MM-32X-AT User Manual V1.00
IO 302	Diamond Systems RUBY-MM-1612 User Manual V1.1
IO 303	Diamond Systems RUBY-MM-416 User Manual V1.1
IO 304	Diamond Systems ONYX-MM-XT User Manual V1.4
IO 305	Diamond Systems QUARTZ-MM User Manual V1.5
IO 306	DM5814/DM6814 User's Manual from Real Time Devices
IO 308	Softing CAN-AC2-104 User Manual Version 4.0
IO 309	MPL PATI Preliminary User Manual

### Connecting Hardware to Screw Terminal Boards

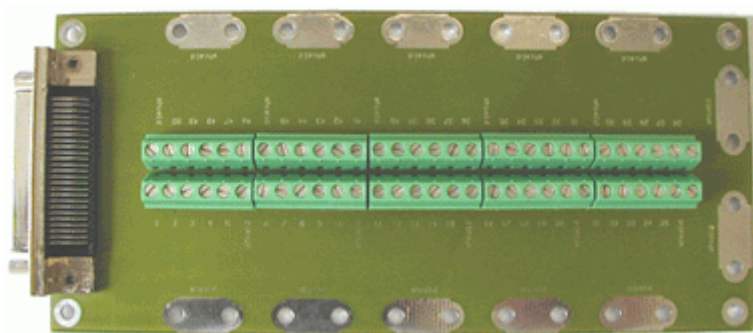
Be extremely careful when making connections to the terminal boards. For your convenience the shipping case includes one or two I/O cables and one or two screw terminal boards (depending on the I/O option) for each I/O option installed in your xPC TargetBox. You can use these parts to quickly and conveniently make the connections between your xPC TargetBox and the hardware you are testing.

---

**Note** Unplug an xPC TargetBox when connecting any I/O cables or screw terminal boards to an xPC TargetBox.

---

- 1 You can find the screw terminal boards in the shipping case at the front of the top compartment.



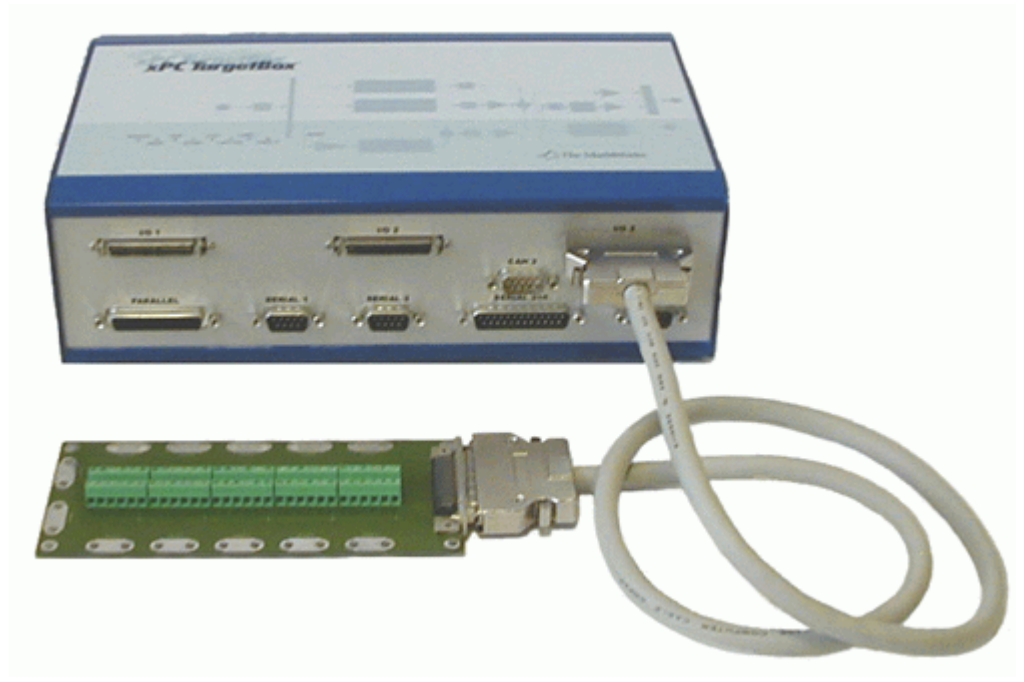
- 2 You can find the I/O cables in the lower compartment of the shipping case.
- 3 Plug one end of the I/O cable into an I/O connector. Press both latches at the plug at the same time and insert the plug. Check that the plug is properly connected to the I/O connector.
- 4 Connect the screw terminal board to the other side of the I/O cable in the same way. Make sure that the screw terminal board does not have any wires connected to it.



---

**Caution** Especially make sure that no physical connections exist between the screw terminal board and the hardware under test, which could possibly be powered up and therefore immediately damage I/O options or the xPC TargetBox.

---



It is good practice to do some loopback testing of an I/O option before connecting it to your real hardware. See Chapter 5, “xPC TargetBox I/O Options Reference”.

The pin layout for each I/O option is provided in tables found in Chapter 5, “xPC TargetBox I/O Options Reference”.



# Regular Use

---

This chapter describes the expected regular use of xPC Target with an xPC TargetBox. For regular use, xPC TargetBox is connected to a separate host PC running xPC Target and a floppy disk drive to boot the xPC TargetBox. This chapter includes the following sections:

xPC Target Software Installation  
(p. 3-3)

How to install the xPC Target software and activate the additional boot modes. This topic also describes how to optionally install the xPC Target Embedded Option so that you can deploy stand-alone GUI applications on a host PC system that does not contain your licensed copy of xPC Target.

xPC TargetBox Hardware  
Installation (p. 3-7)

How to remove the test dongles and connect a floppy disk drive.

Host PC to xPC TargetBox  
Communication (p. 3-14)

How to select RS-232 communication for an easy and inexpensive installation, or select TCP/IP communication for faster data transfer rates and longer connections.

xPC TargetBox Test (p. 3-22)

How to boot the xPC Target kernel on an xPC TargetBox and establish a connection with the host PC. This topic also describes how to test connections and communication between the hostPC and an xPCTargetBox by building,

downloading, and running a simple target application.

## xPC Target Software Installation

Before you can build and download a target application from the host PC to an xPC TargetBox, you need to properly install the required MathWorks software. This section includes the following topics:

xPC Target Software (p. 3-3)	Installation notes for MATLAB, Simulink, Stateflow and Stateflow Coder (optional), Real-Time Workshop, xPC Target, and a third-party C/C++ compiler on the host PC
Activating the Additional xPC TargetBox Boot Modes (p. 3-4)	How to activate DOSLoader and StandAlone modes with an xPC TargetBox, but without xPC Target Embedded Option
Configuring xPC TargetBox Host PC (p. 3-4)	How to configure the xPC TargetBox host PC through xPC Target Explorer
xPC Target Embedded Option Software (p. 3-6)	How to install xPC Target Embedded Option on the host PC

### xPC Target Software

You need to have an active xPC Target license for each xPC TargetBox you purchase, or you need to have one active license for xPC Target Embedded Option with the purchase of two or more xPC TargetBox units.

The xPC Target software is installed entirely on the host PC. Installing software on the xPC TargetBox is not necessary. The xPC Target software includes the following additional boot modes for an xPC TargetBox only:

- DOSLoader — Boot the kernel from the xPC TargetBox flash disk and download a target application from a host PC.
- StandAlone — Create a stand-alone kernel/application and load it on an xPC TargetBox from a flash disk.

For information on installing software on the host computer, see Installation and Configuration in Getting Started with xPC Target.

---

**Note** If you do not have the xPC Target Embedded Option, you need to activate the additional boot modes. See “Activating the Additional xPC TargetBox Boot Modes” on page 3-4.

---

## Activating the Additional xPC TargetBox Boot Modes

If you purchased xPC Target and an xPC TargetBox, but you did not purchase the xPC Target Embedded Option, you need to activate the additional boot modes (DOSLoader and StandAlone).

---

**Note** Even if you do not have an xPC TargetBox, you can activate the additional boot modes and use xPC Target Explorer for xPC TargetBox to create a boot disk. However, if you try to use this boot disk on a target PC that is not an xPC TargetBox, the kernel will not load.

---

- 1 On the host computer, start MATLAB.
- 2 In the MATLAB Command Window, type

```
xpctargetboxinit
```

The additional boot modes are now active in the xPC TargetBox Explorer window for the xPC TargetBox.

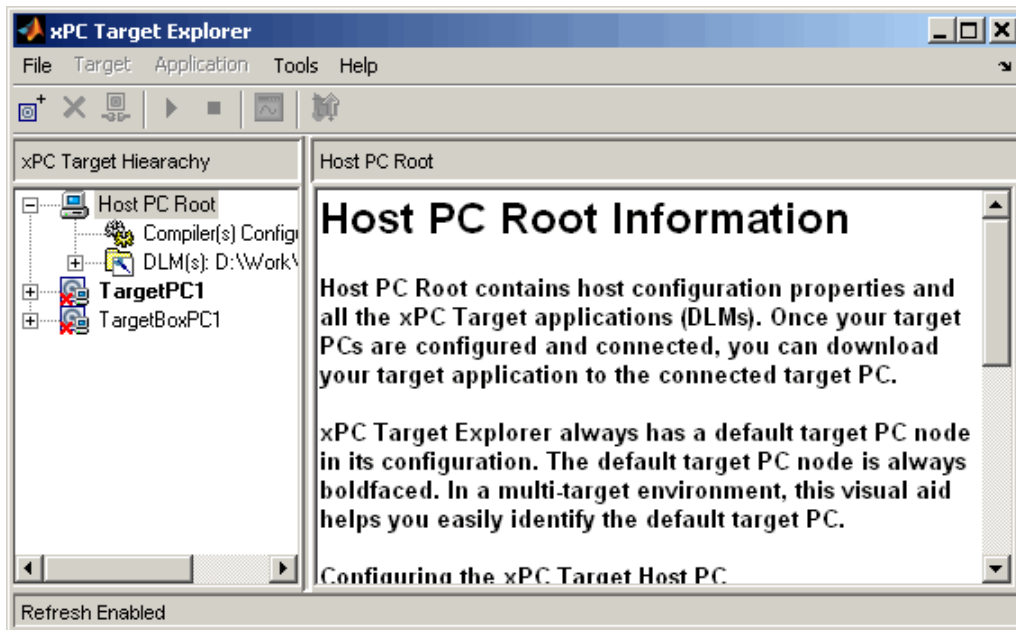
## Configuring xPC TargetBox Host PC

Configure the host PC for your compiler. The following procedure describes how to set up serial communication environment properties through xPC Target Explorer.

- 1 If xPC Target Explorer is not already open, in the MATLAB Command Window, type

```
xpccexplr xpctargetbox
```

The xPC Target Explorer window specific for xPC TargetBox opens.

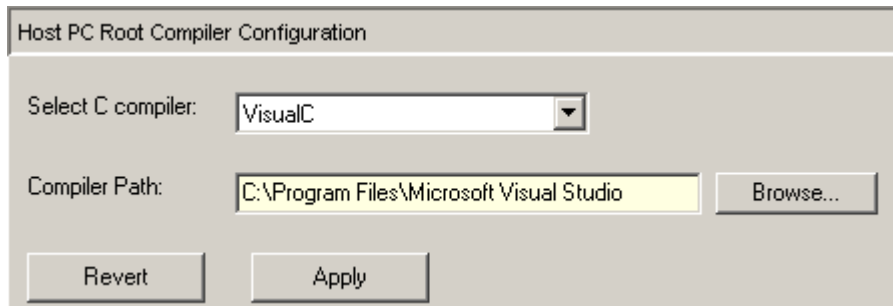


- 2 In the xPC Target Explorer, select the Compiler(s) Configuration node.

In the right pane, the compiler parameters appear.

- 3 At the **Select C Compiler** drop-down list, select the compiler you have installed on the host PC. The examples in this chapter use VisualC.
- 4 Enter the path (or browse) to the compiler for **Compiler Path**. For example,

C:\Program Files\Microsoft Visual Studio



**5** Click **Apply** to apply the changes.

For information on entering and selecting parameters for the xPC TargetBox, see “Host PC to xPC TargetBox Communication” on page 3-14.

---

**Note** xPC Target Explorer dialogs highlight a field and enable the **Revert** and **Apply** buttons when you make changes. To apply changes, click **Apply**. A prompt is displayed if you leave a dialog without first saving changes. If you want the original entry to be displayed, click the **Revert** button and do not click the **Apply** button. If you click **Apply**, you cannot retrieve the original entries.

---

## **xPC Target Embedded Option Software**

You do not need the xPC Target Embedded Option to use an xPC TargetBox. But if you have an xPC Target Embedded Option license, you can use the following additional feature with an xPC TargetBox:

- Deployment — License to deploy an unlimited number of applications developed with xPC Target on any target PC including an xPC TargetBox.

For information on installing software on the host computer, see Installation and Configuration in Getting Started with xPC Target.



## xPC TargetBox Hardware Installation

If you completed the self-test successfully, you have already connected a monitor to your xPC TargetBox. To further use your xPC TargetBox, you need to attach an external floppy disk drive and, optionally, a keyboard and mouse. This section includes the following topics:

Removing Test Dongles (p. 3-7)	How to remove and save the dongles to troubleshoot I/O boards later
Connecting the External Floppy Disk Drive (p. 3-8)	How to connect the external floppy drive and copy files from the host PC to an xPC TargetBox
Connecting Additional Peripherals (p. 3-11)	How to connect a mouse and keyboard to directly interact with the target application running on an xPC TargetBox

### Removing Test Dongles

After successfully running the self-test, you can remove the test dongles from the xPC TargetBox:

- 1 Unplug the AC power cord.

- 2** Press the two latches on the sides of each dongle at the same time and remove the dongle from the connector.



- 3** Store the test dongles in the lower compartment of the shipping case for later use.

If you encounter a problem with an I/O option in the future, you will need these test dongles to narrow down the problem.

## **Connecting the External Floppy Disk Drive**

You use the external floppy disk drive to boot the xPC Target kernel on the xPC TargetBox and, with DOSLoader mode, to copy the xPC Target kernel and target application to the flash disk of the xPC TargetBox. With DOSLoader mode, you can copy the files and then remove the disk drive.

- 1** In the shipping case and from the upper layer, remove the external floppy disk drive. Check it visually for physical damage.

- 2** From the compartment below the disk drive, remove the external disk drive cable.



- 3** Connect the correct end of the cable to the connector on the back end of the external floppy disk drive. Tighten the security screws.
- 4** On the back plate of the xPC TargetBox, locate the external floppy disk drive connector. This special 25-pin port connector is clearly marked. Connect the other end of the external floppy disk drive cable to this connector and tighten the security screws.



**5** Leave the xPC TargetBox unpowered.



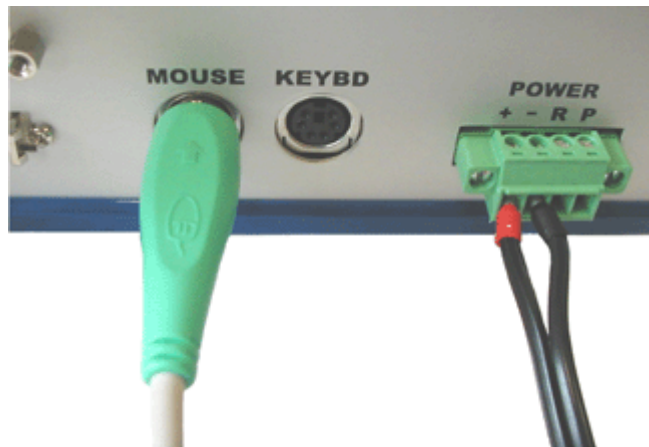
The next step is to connect the xPC TargetBox to a host computer using either a serial or network connection. See “Host PC to xPC TargetBox Communication” on page 3-14.

## Connecting Additional Peripherals

When using the boot modes BootFloppy and DOSLoader, the xPC TargetBox is connected to a host PC. From this host PC you can control and interact with a target application running on an xPC TargetBox.

When using StandAlone mode, there is no connection to the host PC. In this case, you can only interact with an xPC TargetBox through a video monitor, mouse, and keyboard connected directly to the xPC TargetBox.

- 1 Use a mouse with a PS/2 connector and attach it to the back panel of your xPC TargetBox.



- 2 Use a keyboard with a PS/2 connector and attach it to the back panel of your xPC TargetBox.



**3** Attach the video monitor VGA connector to the back of the xPC TargetBox.

With a mouse and keyboard attached to your xPC TargetBox, you can use the target command-line interface to interact with target applications. For information about the target PC command-line interface, see *Using the Target PC Command-Line Interface* in the *xPC Target User's Guide*.



## Host PC to xPC TargetBox Communication

Before you can create and run a target application, you need to set up the connection between your host PC and your xPC TargetBox. You can use either serial or network communication. This section includes the following topics:

Hardware for Serial Communication (p. 3-14)	How to connect a null modem cable
Environment Properties for Serial Communication (p. 3-14)	How to enter the host PC COM port and baud rate for serial communication
Hardware for Network Communication (p. 3-17)	How to connect to a LAN or which crossover cable to use
Environment Properties for Network Communication (p. 3-17)	How to enter the IP address and network information for an xPC TargetBox

### Hardware for Serial Communication

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

Null modem cable — Connect the host PC to an xPC TargetBox with the null modem cable supplied by The MathWorks. This cable was shipped with the xPC Target software. You can use either the COM1 or COM2 port on the host PC, and either the SERIAL 1 or SERIAL 2 port on the xPC TargetBox.

### 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 need to set these properties before you can build and download a target application.

After you install xPC Target and activate the additional boot modes, you can set the environment properties for the host PC and xPC TargetBox. See “xPC Target Software Installation” on page 3-3.



- 1 In the MATLAB Command Window, type

```
xpcexplr xpctargetbox
```

The xPC Target Explorer window specific for xPC TargetBox opens. xPC Target Explorer always has a default target PC node in its configuration. The default target PC node is always boldfaced. In a multitarget environment, this visual aid helps you easily see the target PC you are working with.

- 2 Right-click Host PC Root.
- 3 Select Add TargetBox.

A target PC node, named TargetBoxPC1, appears in the **xPC Target Hierarchy**, at the same level as the HostPC node and under the default TargetPC1.

- 4 In the xPC Target Explorer, expand the xPC TargetBox node.

A Configuration node appears. Under this are nodes for Communication, Settings, and Appearance. The parameters for the xPC TargetBox node are grouped in these categories.

- 5 Select Communication.

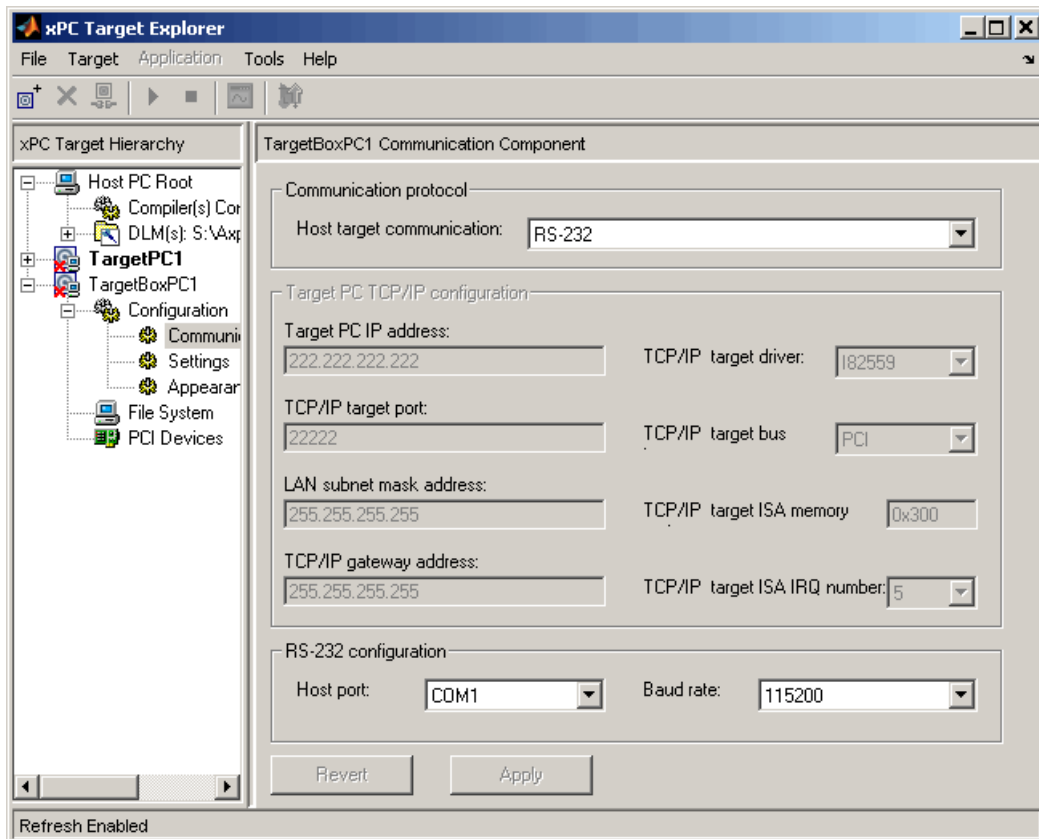
The **Communication Component** pane appears to the right.

- 6 From the **Host target communication** list, select RS232.

The pane changes to one that contains only those parameters pertinent to serial communication.

- 7 From the **Host port** list, select either COM1 or COM2 for the connection on the host PC. xPC Target determines the COM port you use on the xPC TargetBox automatically.
- 8 From the **Baud rate** list, select the baud rate for the serial connection between the host PC and this xPC TargetBox. The default is 115200.

Click the **Apply** button to apply the changes.



After making changes to the xPC Target environment, you must recreate the target boot disk and rebuild the target application from the Simulink model.

For more information on the xPC Target environment, see Software Environment and Demos in the xPC Target User's Guide.

Your next task is to create a target boot disk. See "Creating a Target Boot Disk" on page 3-22.

## Hardware for Network Communication

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

- 1 When using the xPC TargetBox with TCP/IP, you must have a network adapter card correctly installed on the host PC. The xPC TargetBox has an Intel 82559ER Ethernet controller.
- 2 Connect the host and target computers with an unshielded twisted pair (UTP) cable to your local area network (LAN). On the front panel of the xPC TargetBox, locate the 10/100 MB/s Ethernet connector. Plug the target side of the Ethernet cable into the Ethernet plug of the xPC TargetBox.

You can also directly connect your computers. Use a crossover UTP cable with RJ-45 connectors.

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.

## 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 must set these environment properties before you can build and download a target application.

After you install xPC Target and activate the additional boot modes, you can set the environment properties for the host PC and xPC TargetBox. See “xPC Target Software Installation” on page 3-3.

- 1 In the MATLAB Command Window, type

```
xpcexplr xpctargetbox
```

The xPC Target Explorer window opens.

- 2 In the xPC Target Explorer, expand an xPC TargetBox node.

A Configuration node appears. Under this are nodes for Communication, Settings, and Appearance. The parameters for the xPC TargetBox node are grouped in these categories.

**3** Select Communication.

The **Communication Component** pane appears to the right.

**4** From the **Host target communication** list, select TcpIp.

The pane changes to one that contains only those parameters pertinent to network communication.

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

- **Target PC TcpIp address** — This is the IP address for your xPC TargetBox. An example of an IP address is 192.168.0.10.
- **LAN subnet mask address** — This is the Subnet Mask address of your LAN. An example of a Subnet Mask address is 255.255.0.0.

**6** Optionally enter the following properties, depending on your specific circumstances:

- **TcpIp target port** — 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 relevant on the xPC TargetBox. If necessary, you can change this property value to any value higher than 20000 and less than 65536.
- **TcpIp gateway address** — This property is set by default to 255.255.255.255. This means that you do not use a gateway to connect to your xPC TargetBox.

If you communicate with the xPC TargetBox from within your LAN, you might not need to define a gateway and change this setting.

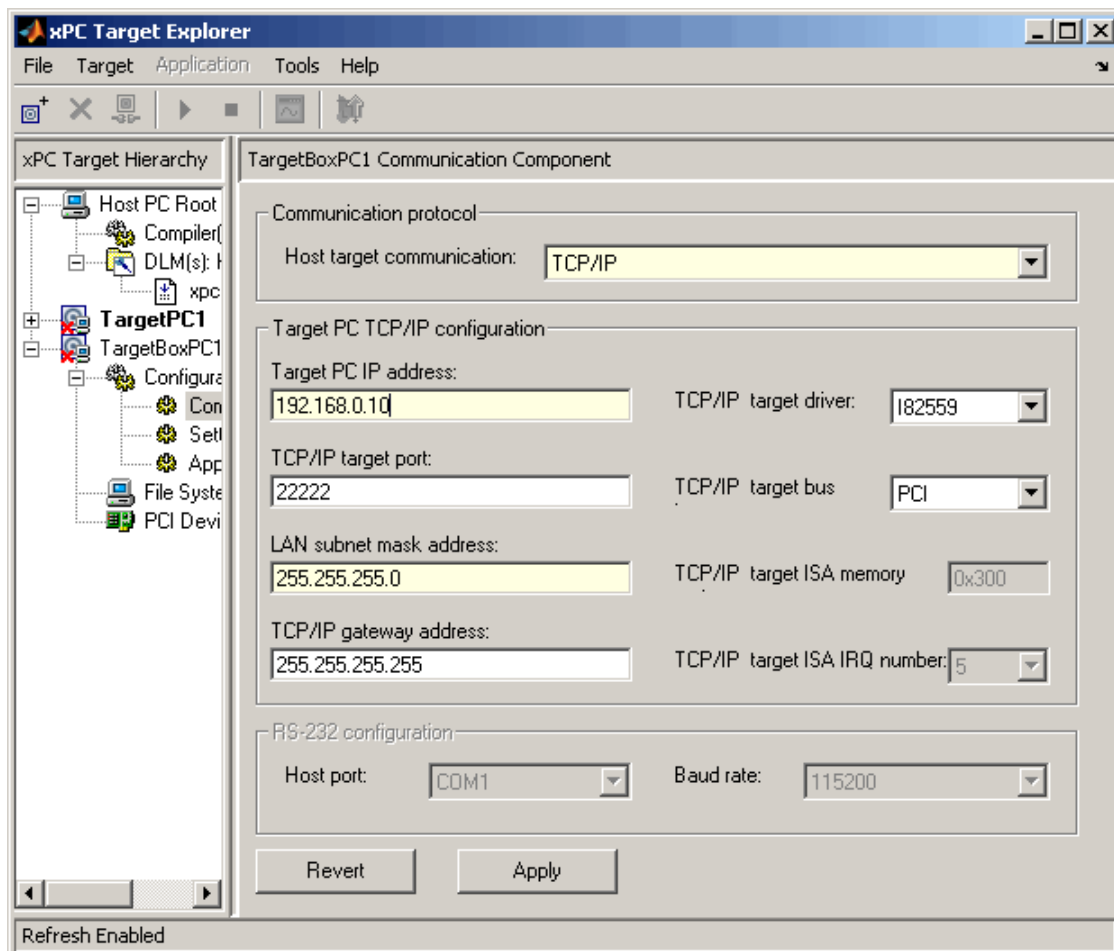
If you communicate from a host PC located in a LAN different from your xPC TargetBox, you need to define a gateway and enter its IP address.

This is especially true if you want to work over the Internet. Ask your system administrator for the IP address of the appropriate gateway.

- 7 Enter the following properties specific for the Ethernet card on your xPC TargetBox:
  - **TcpIp target driver** — For xPC TargetBox, the software already chooses the appropriate driver.
  - **TcpIp target bus type** — For xPC TargetBox, the software already chooses the appropriate bus type.
  - **TcpIp target ISA memory port** and **TcpIp target ISA IRQ number** — For xPC TargetBox, these properties are disabled.

The following figure illustrates the **Configuration Component** pane for a network connection.

xPC Target Explorer updates the properties as you change them.



After making changes to the xPC Target environment, you must recreate the target boot disk and rebuild the target application from the Simulink model.

For more information on the xPC Target Environment, see Software Environment and Demos in the xPC Target User's Guide.

Your next task is to create a target boot disk. See "Creating a Target Boot Disk" on page 3-22.

## xPC TargetBox Test

The target boot disk includes the xPC Target kernel specific for either serial or network communication. If you select StandAlone mode, the flash disk also includes the target application. This section includes the following topics:

Creating a Target Boot Disk (p. 3-22)	How to use the <code>xpcexplr</code> command to open a dialog box and create a boot disk
Booting an xPC TargetBox (p. 3-24)	How to check xPC TargetBox connections to peripherals and power supply
Testing the Installation (p. 3-27)	How to run the <code>xpctest</code> command
Test 1, Ping Target System Standard Ping (p. 3-29)	How to test network communication with a standard ping command
Test 2, Ping Target System xPC Target Ping (p. 3-30)	How to test either network or serial communication with the host PC
Test 3, Reboot Target Using Direct Call (p. 3-31)	How to test rebooting an xPC TargetBox from the host PC
Test 4, Build and Download Application (p. 3-31)	How to test downloading and running a target application on an xPC TargetBox

### Creating a Target Boot Disk

You use the target boot disk to load and run the xPC Target kernel on the xPC TargetBox. After you make changes to the xPC Target environment properties, you need to create or update a target boot disk.

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

- 1 If the xPC Target Explorer is not open, open it now. At the MATLAB Command Window, type

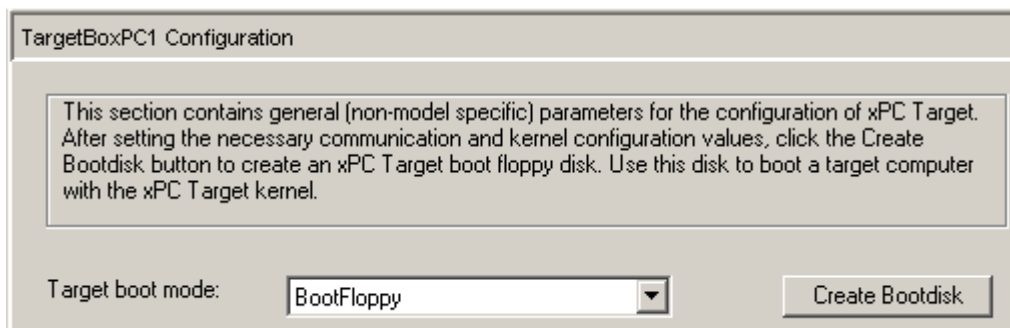
```
xpcexplr xpctargetbox
```



- 2 In the **xPC Target Explorer xPC Target Hierarchy** pane, select the xPC TargetBox Configuration node. For example, select the Configuration node for TargetBoxPC1.

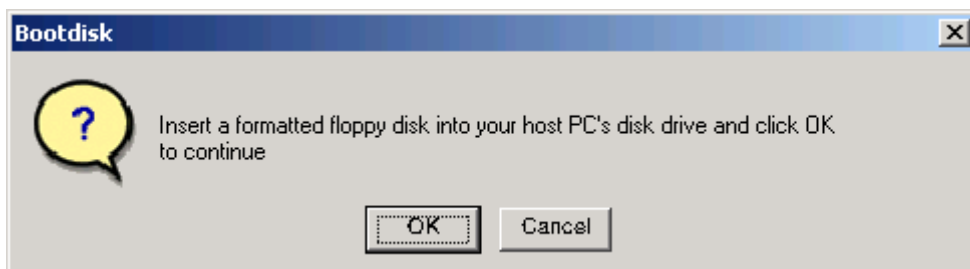
A **TargetBoot** pane appears in the rightmost pane.

- 3 From the **Target boot mode** list, select a boot disk mode. For example, select BootFloppy.



- 4 Click the **Create Bootdisk** button.

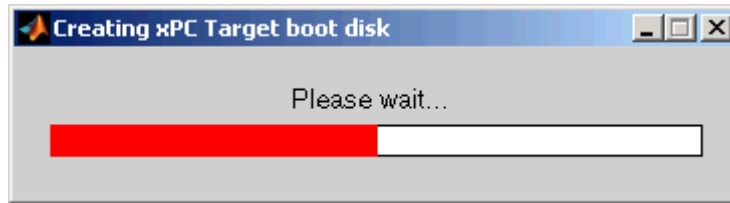
The following message box opens.



- 5 Insert a formatted 3.5 inch floppy disk into the host PC disk drive, and then click **OK**.

All data on the disk will be erased as xPC Target writes the xPC Target kernel and other required files to the 3.5 inch disk.

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



- 7 When the 3.5 inch disk drive stops, remove the 3.5 inch disk.
- 8 Remove the target boot disk from the host PC disk drive and insert it into the external disk drive connected to your xPC TargetBox.

Your next task is to test your installation. See “Booting an xPC TargetBox” on page 3-24.

## **Booting an xPC TargetBox**

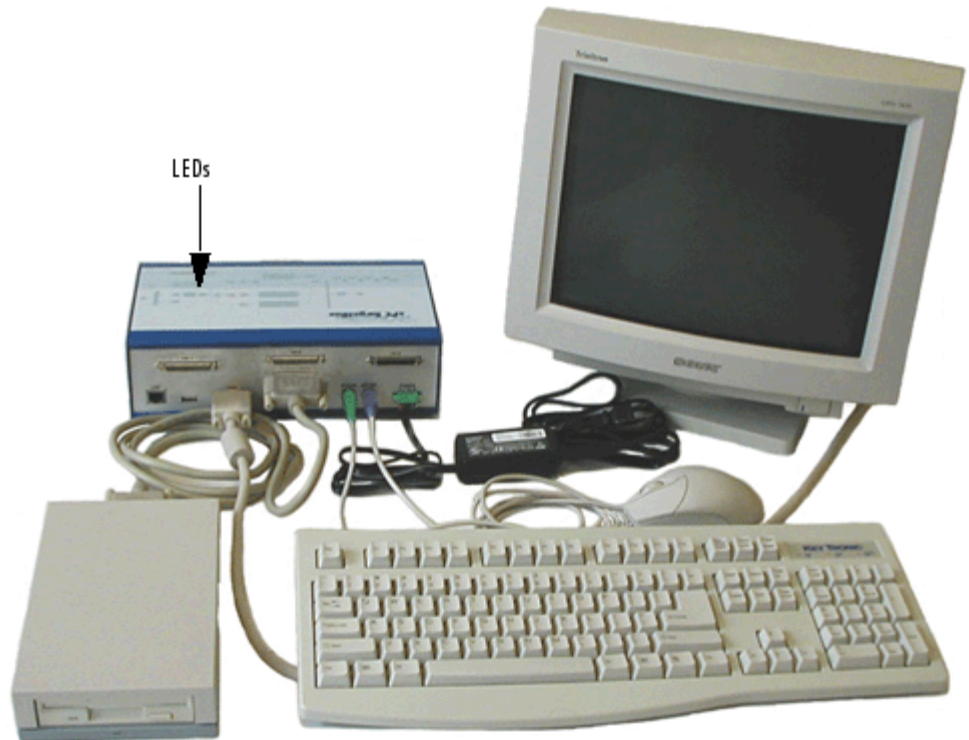
When using an xPC TargetBox with a floppy disk drive, you can leave the self-test application active on the flash disk. This works because the default BIOS setting is to boot A:\ (floppy disk) before C:\ (flash disk).

After you create a boot disk from the xPC Target Explorer window, you are ready to power up the xPC TargetBox again. This is a good time to check the xPC TargetBox connections.

- 1 For the xPC TargetBox, check that the external AC power supply adapter is properly connected to the xPC TargetBox, the external floppy drive is properly connected, the monitor is connected and powered up, an Ethernet or RS-232 cable is properly connected, and the xPC TargetBox is still unpowered.
- 2 Insert an xPC Target boot disk into the external drive.

- 3 Plug in the AC power cord.

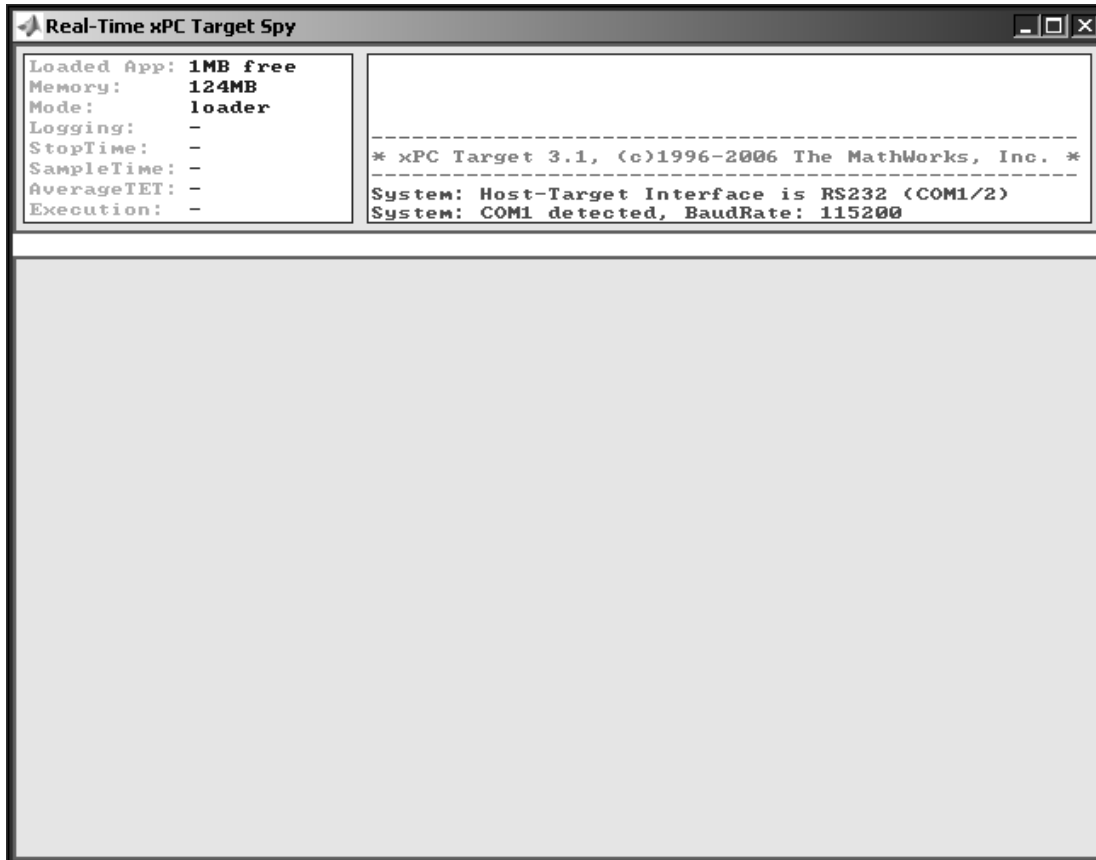
xPC TargetBox starts immediately. On the top of the xPC TargetBox, check that the Reset LED and then the Power LED turn on.



- 4 After the BIOS loads (time varies with xPC TargetBox model), the floppy disk drive is accessed (verify by observing the green LED on the front of

the external floppy disk drive and listening for an audible sound from the drive).

The xPC TargetBox loads the xPC Target kernel on the disk. As soon as the entire image has been loaded, the xPC Target kernel starts executing on the xPC TargetBox.



The xPC TargetBox and the loaded xPC Target kernel are now ready to accept commands from the host computer through either the RS-232 or Ethernet connection.

---

**Note** As long as the xPC TargetBox is powered on, you can later reboot the xPC TargetBox by pressing the RESET button on the top of the xPC Targetbox or by unplugging and replugging the AC power cord.

---

Your next step is to test the installation. See “Testing the Installation” on page 3-27.

## Testing the Installation

xPC Target uses a test script to test the entire installation. Use this test to troubleshoot connection and communication problems between your host PC and an xPC TargetBox. This test checks both the host computer xPC Target setup and the xPC TargetBox by building, downloading, and running a simple test Simulink model. This procedure assumes that you have set environment settings with xPC Target Explorer. See “Environment Properties for Serial Communication” on page 3-14 and “Environment Properties for Network Communication” on page 3-17.

After you install the xPC Target software, set the environment settings, and create a target boot disk, you can test your installation. This task assumes that you have performed the task in “Booting an xPC TargetBox” on page 3-24.

- 1** In the MATLAB Current Directory window, select a MATLAB current directory outside 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.

---

- 2** 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 RS-232 communication, the first test is skipped.

```
### xPC Target Test Suite 3.1
### Host-Target interface is: RS232 COM1
### Test 1, Ping target system using standard ping: ... SKIPPED
### Test 2, Ping target system using xpctargetping: ... OK
### Test 3, Reboot target using direct call: ..... OK
### Test 4, Build and download xPC Target application: ... OK
### Test 5, Check host-target communication for commands: .. OK
### Test 6, Download xPC Target application using OOP: ... OK
### Test 7, Execute xPC Target application for 0.2s: ... OK
### Test 8, Upload logged data and compare with simulation:. OK
### Test Suite successfully finished
```

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

- “Test 1, Ping Target System Standard Ping” on page 3-29
- “Test 2, Ping Target System xPC Target Ping” on page 3-30
- “Test 3, Reboot Target Using Direct Call” on page 3-31
- “Test 4, Build and Download Application” on page 3-31

If all the subtests are successful, the host computer and xPC TargetBox are properly set up for regular use. Your next steps are to

- Connect the xPC TargetBox to the hardware you are testing through the I/O connector ports. See “Connecting Hardware to an xPC TargetBox” on page 2-18.
- Build and download a target application to the xPC TargetBox. See Basic Tutorial in Getting Started with xPC Target.

## 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 xPC TargetBox:

```
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 from 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 might be with your network cables.

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

**Ping command fails, but cables are okay** — If the cables are okay, the problem might be that you entered an incorrect property in xPC Target Explorer.

To solve this problem, in the MATLAB Command Window on the host PC, type

```
xpcexplr
```

Check the **TargetIPTargetAddress**, **TcpIpSubNetMask**, and **TcpIGatewayAddress** settings, and create a new boot floppy disk. On the xPC TargetBox, reboot with the corrected boot floppy disk.

**Ping succeeds, but test 1 with the command `xpctest` fails** — The problem might be that you have incorrect IP and gateway addresses entered in xPC Target Explorer.

To solve this problem, in the MATLAB Command Window on the host PC, type

```
xpcexplr
```

Enter the correct addresses. Recreate the target boot disk by inserting a floppy disk into the host PC disk drive and then clicking the **Create Bootdisk** button.

If you still cannot solve your problem, see “If You Still Need More Help” on page 3-32.

## Test 2, Ping Target System xPC Target Ping

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

- 1 In the MATLAB Command Window, type

```
tg=xpctarget.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 these messages, the problem might be that you have a bad target boot disk.

To solve this problem, create another target boot disk with a new floppy disk. See “xPC TargetBox Test” on page 3-22.

If you still cannot solve your problem, see “If You Still Need More Help” on page 3-32.



## Test 3, Reboot Target Using Direct Call

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

- 1 In the MATLAB Command Window, type

```
xptest 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.

If you still cannot solve your problem, see “If You Still Need More Help” on page 3-32.

## 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 xPC TargetBox. This error message is sometimes displayed even if the target screen shows the loader is ready.

- 3 Open xPC Target Explorer. Type

```
xpcexplr xpctargetbox
```

- 4 Check the path to the C compiler. A common error when creating a target application is setting the path to the C compiler incorrectly.

If you still cannot solve your problem, see “If You Still Need More Help” on page 3-32.

**If You Still Need More Help**

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

**Internet**

To contact The Mathworks  
Technical Support, use this form  
[http://www.mathworks.com/contact\\_TS.html](http://www.mathworks.com/contact_TS.html)

**Telephone**

508-647-7000  
Ask for Technical Support.

# Advanced Use

---

This chapter describes some alternate ways that you can use xPC TargetBox. For example, your xPC TargetBox can boot from devices other than a floppy disk. You can also disconnect it from a host computer for stand-alone operation. In addition, you can copy files to an xPC TargetBox that is set up for running stand-alone applications, using File Transfer Protocol (FTP). This chapter includes the following sections:

xPC TargetBox Library (p. 4-2)	Describes the Simulink blocks that represent drivers for the xPC TargetBox I/O options
DOSLoader Mode (p. 4-6)	Describes how to use the DOSLoader mode to boot the kernel from flash memory
StandAlone Mode (p. 4-12)	Describes how to use the StandAlone mode to boot the kernel and the target application from flash memory without a connection to a host PC
FTP File Transfer (p. 4-15)	Describes how to copy files from a host PC to an xPC TargetBox using an Ethernet connection
Booting xPC TargetBox to DOS (p. 4-22)	Describes how to boot the xPC TargetBox into DOS

## **xPC TargetBox Library**

The xPC TargetBox library consists of Simulink blocks that represent the drivers for the xPC TargetBox I/O options, LEDs, and watchdog circuitry. This section includes the following topics:

Drivers in the xPC TargetBox Library (p. 4-2)	Description of the drivers from the xPC Target library. These drivers are conveniently grouped into an xPC TargetBox library.
Using the Panel LEDs (p. 4-4)	Description of how control the two LEDs on the xPC TargetBox using driver blocks
Using the Watchdog Timer (p. 4-5)	Description of how to perform a system reset when a programmable timeout occurs

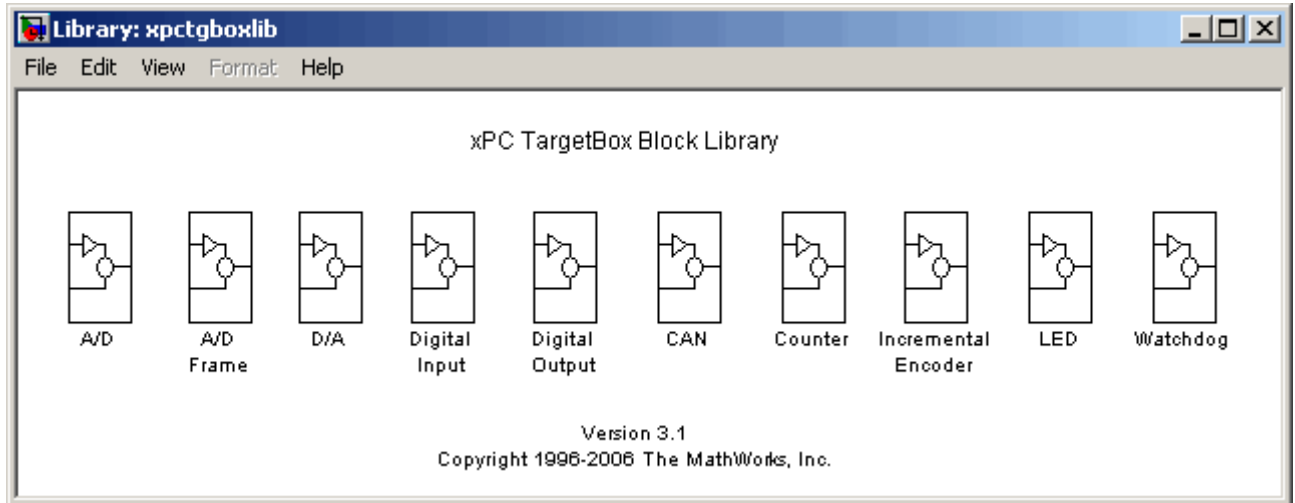
### **Drivers in the xPC TargetBox Library**

The drivers in the xPC TargetBox library are links to drivers from the various function groups in the xPC Target driver library. Use these blocks in the same way as you would use the original driver blocks:

- 1 In the MATLAB Command Window, type

```
xpctgboxlib
```

The Library: xpctgboxlib window opens.



- 2 Drag and drop a driver block into your Simulink model.

Except for the LED and Watchdog blocks, the driver blocks in the xPC TargetBox library are links to drivers in the xPC Target library. For information about a driver block,

- Click the **Help** button in the Block Parameters dialog box.
- For recently added drivers, use the updated xPC Target I/O Reference on the xPC Target Product News page:

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

---

**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.

In particular, hardware input blocks in the xPC TargetBox library (blocks that acquire data from hardware) are affected by this change.

---

### Using the Panel LEDs

Each xPC TargetBox has two LEDs that you can control. These LEDs are labeled USER 1 and USER 2 on an xPC TargetBox.

From the xPC TargetBox library (`xpctgbox.lib`), drag an LED block into your Simulink model. You can have only one block of this type in any model.

### Driver Block Parameters

**LED vector** — Enter either 1 and 2. This driver allows the selection of individual LEDs in any order. The number of elements defines the number of LEDs used. For example, to control the USER 1 and USER 2 LEDs, enter

```
[1 2]
```

**Reset vector** — Enter 0 or 1 for each LED. This parameter controls the behavior at model termination. A value of 1 causes the corresponding LED to be reset to the value specified in **Initial value vector**. A value of 0 causes the corresponding LED to be left at the most recent value set while the model was running. The **Reset vector** should either be a scalar value or the same length as the LED vector. For scalar values, the block performs a scalar expansion and assumes the same **Reset vector** value for each element in the vector.

**Initial value vector** — Enter 0 or 1 for each LED. This parameter specifies the initial value (0 or 1) to which the LED should be set between the time the model is downloaded and the time it is started. The **Initial vector value** should be the same length as the LED vector. If you do not have the same vector length, the block performs a scalar expansion and assumes the same **Initial vector value** value for each element in the vector.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

## Using the Watchdog Timer

The xPC TargetBox provides a watchdog that you can program to perform a system reboot when a programmable timeout occurs. The timeout interval can range from 1 to 255 seconds with one-second resolution, or from 1 to 255 minutes with one-minute resolution. When the timeout expires, an IRQ15 interrupt is asserted. With the **Reboot upon expiration of watchdog** parameter, you can specify that the system reboot at this time. At most, one block of this type can be included in any model.

### Driver Block Parameters

**Timeout units** — From the list, select either seconds or minutes. These are the units for the time.

**Timeout interval** — Enter a number between 1 and 255.

**Show reset port (R)** — Select this check box to display an input port labeled R. A signal connected to this port resets the watchdog whenever its value exceeds 0.5.

**Reset upon keyboard activity** — Select this check box to reset the watchdog whenever there is keyboard activity.

**Reset upon mouse activity** — Select this check box to reset the watchdog whenever there is mouse activity.

**Reboot upon expiration of watchdog** — Select this check box to reboot the xPC TargetBox system when the watchdog timer expires. When the timeout expires, an IRQ15 interrupt is always asserted, regardless of the state of this check box. If you choose not to reboot the system when the watchdog timer expires, you might want to write your own procedure to handle the IRQ15 assertion.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

## DOSLoader Mode

DOSLoader mode allows you to copy the xPC Target kernel to the xPC TargetBox flash disk, remove the floppy disk drive, and then boot the xPC Target kernel from the flash disk of the xPC TargetBox. The target application is still downloaded from the host PC.

Use this mode for applications where an xPC TargetBox is not easily accessible or where the constant presence of a floppy disk drive is not possible or easy to maintain. This section includes the following topics:

Updating Environment Properties and Creating a Boot Disk (p. 4-7)	How to select DOSLoader mode in xPC Target Explorer
Copying the Kernel to Flash Memory (p. 4-8)	How to copy the xPC Target kernel to the flash disk on an xPC TargetBox and then start the kernel running
Creating a Target Application (p. 4-10)	How to create, download, and run a target application from a host PC

---

**Note** DOSLoader mode is included with xPC TargetBox when using xPC Target Explorer. To use DOSLoader mode with another target PC, you need to use the xPC Target Embedded Option. See Embedded Option in the xPC Target User's Guide. See "Activating the Additional xPC TargetBox Boot Modes" on page 3-4 on "Activating the Additional xPC TargetBox Boot Modes" on page 3-4 for instructions on how to activate DOSLoader mode even if you do not have xPC Target Embedded Option.

---

The following topics assume that you have previously and successfully created a target application with the BootFloppy mode. Although this is not necessary, it helps ensure that your xPC TargetBox is functioning properly before you try to create a kernel that boots from DOS.



## Updating Environment Properties and Creating a Boot Disk

This procedure assumes you have a proper serial or network communication between your host computer and an xPC TargetBox.

- 1 On the host computer, start MATLAB.
- 2 In the MATLAB Command Window, type
 

```
xpceplr xpctargetbox
```

xPC Target Explorer opens.
- 3 In the xPC Target Explorer **xPC Target Hierarchy** pane, select the xPC TargetBox Configuration node.
- 4 From the **Target boot mode** list, select DOSLoader.
- 5 Click **Create Bootdisk**.

A message box opens with the following message.

```
Insert a formatted floppy disk into your host PC disk drive and
click OK to continue.
```

- 6 Insert a floppy disk; then click **OK**.

The files checksum.dat, xpcsgo1.rtb (serial) or xpctgo1.rtb (TCP/IP), xpcboot.com, and autoexec.bat are copied to the disk.

With DOSLoader mode, the correct \*.rtb file is added to the DOS disk according to the options specified in the following table.

<b>xPC Target Environment</b>	<b>HostTargetComm: RS-232</b>	<b>HostTargetComm: TCP/IP</b>
<b>TargetScope:</b> Disabled	xpcston.rtb	xpctton.rtb
<b>TargetScope:</b> Enabled	xpcsgon.rtb	xpctgon.rtb

The numeric value of  $n$  corresponds to the maximum model size. This value is either 1, 4, or 16 megabytes. The default value for  $n$  is 1, or a 1-megabyte maximum model size.

## Copying the Kernel to Flash Memory

One method for transferring the kernel files from a host PC to an xPC TargetBox is to use an external floppy disk drive.

After you create the boot disk with the kernel files on a host PC, you can then copy those files from the boot disk to the flash disk. See “Updating Environment Properties and Creating a Boot Disk” on page 4-7:

- 1 If there is a floppy disk in the xPC TargetBox external disk drive, remove it. On the xPC TargetBox, press the **Reset** button.
- 2 When you see the message for loading DOS on the xPC TargetBox monitor, press **Ctrl+C** repeatedly on the xPC TargetBox keyboard until you see the message

```
Terminate batch file 'C:\AUTOEXEC.BAT' (Yes/No/All)?
```

If you press **Ctrl+C** too many times, you might see a DOS prompt, such as the following.

```
c:\tests
```

Note that if you have a system with the QuickBoot capability, start pressing **Ctrl+C** repeatedly when you see the blinking cursor.

---

**Note** If you press **Ctrl+C** too early, you might enter the BIOS. Press the **Esc** key to exit the BIOS and continue. Do not save any changes.

---

- 3 If you get the `Terminate batch file` message, type `Y`.

The boot process is stopped and a DOS prompt is displayed.

If you are already at the DOS prompt, go to the next step.

**4** Insert the boot disk with the xPC Target kernel into the external floppy disk drive.

**5** Type

```
copy a:\xpcsgo1.rtb c:\work
copy a:\xpcboot.com c:\work
copy a:\autoexec.bat c:\work
```

**6** If you want the kernel to run when you press the **Reset** button on your xPC TargetBox

**1** Copy `c:\autoexec.bat` to a backup file. For example,

```
copy c:\autoexec.bat c:\autobak.bat
```

**2** Copy the file `c:\autoexec.wrk` to `c:\autoexec.bat`. For example,

```
copy c:\autoexec.wrk c:\autoexec.bat
```

Alternatively, edit the file `c:\autoexec.bat` to include the following lines. Adding these commands to `c:\autoexec.bat` directs the system to execute the `autoexec.bat` file located in `c:\work`.

```
cd c:\work
autoexec
```

---

**Note** Do not confuse `c:\work\autoexec.bat` with `c:\autoexec.bat`. The file `c:\work\autoexec.bat` includes the command `xpcboot.com` to start the xPC Target kernel.

The file `c:\autoexec.bat` includes the files you want the system to execute when the system starts up.

---

**7** Remove the floppy disk, and then, on the xPC TargetBox, press the **Reset** button.

---

**Note** If you get the error message

```
Cannot boot RTTarget-32 from Virtual 8086 mode.
```

edit the `config.sys` file to comment out or remove the line that loads the high memory driver.

```
REM DEVICE=C:\DROS\EMM386.EXE DPMI=OFF FRAME=NO
```

---

Note that xPC TargetBox does not supply a `config.sys` file. If you get this message, it means that you manually added a `config.sys` file to the system. You should know the location of this file.

The sequence of calls during the boot process is

- 1 `c:\autoexec.bat`
- 2 `c:\work\autoexec.bat`
- 3 `c:\work\xpcboot.com`
- 4 `c:\work\xpcsgo1.rtb`

### Creating a Target Application

For DOSLoader mode, a target application is created on a host PC and downloaded to your xPC TargetBox.

After you set the Simulink and Real-Time Workshop parameters for code generation with xPC Target in your Simulink model, you can use xPC Target with DOSLoader mode to create a target application.

- 1 In the MATLAB Command Window, type the name of a Simulink model. For example, type

```
xpc_osc3
```

A Simulink window opens with the model.

- 2** With the Simulink window selected, press **Ctrl+B**.

Alternatively, from the **Tools** menu, point to **Real-Time Workshop**, and then click **Build Model**.

- 3** Real-Time Workshop and xPC Target create a target application and download it to your xPC TargetBox.

## StandAlone Mode

StandAlone mode combines the target application with the kernel and boots them together on the xPC TargetBox from flash memory. The host PC does not need to be connected to the xPC TargetBox. No boot disk is required for the StandAlone mode.

Use this mode for applications where an xPC TargetBox is not easily accessible to a host PC and where the constant presence of a floppy disk drive is not possible or easy to maintain. Note, you cannot build a 16 MB target application to run in StandAlone mode.

This section includes the following topics:

Updating Environment Properties (p. 4-12)	How to select StandAlone mode in xPC Target Explorer
Creating a Kernel/Target Application (p. 4-13)	How to create a stand-alone application on the host PC

---

**Note** StandAlone mode is included with xPC TargetBox when using xPC Target Explorer. To use StandAlone mode with another target PC, you need to use the xPC Target Embedded Option. See the xPC Target documentation. See “Activating the Additional xPC TargetBox Boot Modes” on page 3-4 for instructions on how to activate StandAlone mode even if you do not have xPC Target Embedded Option.

---

The following topics assume that you have previously and successfully created a target application with the BootFloppy mode. Although this is not necessary, it helps ensure that your xPC TargetBox is functioning properly before trying to create a stand-alone application.

### Updating Environment Properties

xPC Target uses the environment properties to determine what files to create for the various target boot modes. This procedure assumes you have a proper serial or network communication between your host computer and an xPC TargetBox.

- 1 On the host computer, start MATLAB.
- 2 In the MATLAB Command Window, type

```
xpcexplr xpctargetbox
```

xPC Target Explorer opens.
- 3 In the xPC Target Explorer **xPC Target Hierarchy** pane, select a target PC Configuration node.
- 4 From the **Target boot mode** list, select StandAlone.

The build process is ready to create a stand-alone kernel/target application.

---

**Note** For StandAlone mode, you do not create an xPC Target boot disk. Instead, you copy files created from the build process to the flash disk.

---

## Creating a Kernel/Target Application

Use xPC Target with StandAlone mode to create a combined kernel and target application with utility files. A combined kernel and target application allows you to disconnect your xPC TargetBox from a host PC and run stand-alone applications.

After you set the Simulink and Real-Time Workshop parameters for code generation with xPC Target in your Simulink model, you can use xPC Target with StandAlone mode to create a target application.

- 1 In the MATLAB Command Window, type the name of a Simulink model. For example, type

```
xpc_osc3
```

A Simulink window opens with the model.

- 2 From the **Tools** menu, point to **Real-Time Workshop**, and then click **Build Model**.

Real-Time Workshop and xPC Target create a directory `xpc_osc3_xpc_emb` with the following files:

```
xpc_osc3.rtb  
xpcboot.com  
autoexec.bat
```

In preparation for setting up the xPC TargetBox to work as a stand-alone system, you need to transfer the kernel/target application files from host PC to the xPC TargetBox. One method for transferring the files is to transfer your files to the target PC flash memory using the File Transfer Protocol (FTP). See “FTP File Transfer” on page 4-15 for details.



## FTP File Transfer

You can create larger target applications (up to 16 MB) that do not fit on a floppy disk. For these applications you can boot the xPC TargetBox in DOSLoader mode and transfer the target application from the host PC (serial or network communication), or you can copy stand-alone files to an xPC TargetBox using a network connection and File Transfer Protocol (FTP).

Another application for FTP transfer is when the xPC TargetBox is in a rugged environment where you do not want to use a mechanical disk drive. In this case, you can remove the floppy disk drive and attach a crossover Ethernet cable from an xPC TargetBox to a host PC.

This section includes the following topics:

Directories and Files on the xPC TargetBox (p. 4-15)	Description of the directories and files on the flash disk
Booting xPC TargetBox with FTP Server (p. 4-17)	Description of how to edit the file <code>autoexec.bat</code> to boot the xPC TargetBox so that the FTP server is running
Using FTP Commands (p. 4-18)	Description of the common FTP commands you enter from the host PC
Copying Files with DOS and FTP (p. 4-18)	Description of how to use FTP to copy stand-alone applications from a host PC to an xPC TargetBox

### Directories and Files on the xPC TargetBox

You need these files to run the FTP server with the network card in the xPC TargetBox. Copy the `autoexec.*` files to `autoexec.bat` as appropriate.

Note, if you corrupt any of these files, contact The MathWorks directly for help.

<b>Directory or Filename</b>	<b>Description</b>
kernel.sys	FreeDOS
command.com	FreeDOS
autoexec.bat	Run current version of autoexec.bat.
autoexec.tst	Run the xPC Target kernel and self-test.
autoexec.wrk	Run a target application.
autoexec.dos	Boot into DOS.
autoexec.ftp	Start the FTP server.
FDOS <DIR>	FreeDOS
FTP <DIR>	Backup files for the FTP server
INTEL <DIR>	Drivers for the Ethernet connection
TESTS <DIR> dostst.exe xpcboot.com xpctbtst.rtb  autoexec.bat	Files to run a self-test for an xPC TargetBox and I/O boards. Do not confuse c:\tests\autoexec.bat with c:\autoexec.bat. c:\tests\autoexec.bat loads and runs the custom target application or self-test. The commands in autoexec.tst call c:\tests\autoexec.bat.
WORK <DIR> xpcboot.com application.rtb autoexec.bat	Files created from building a stand-alone application on the host PC and copied to an xPC TargetBox
ftpbin.exe telpass.exe config.tel password.tel	Files to start the FTP server (ftpbin.exe), set IP address (config.tel), change password (telpass.exe), and contain generated password information (password.tel)
unldftp.bat bootxpc.exe loadxpc.bat	Unload FTP server (unldftp.bat), and unload FTP server with Ethernet drivers (bootxpc.exe, loadxpc.bat)
reboot.exe	Reboot the xPC TargetBox

## Booting xPC TargetBox with FTP Server

An xPC TargetBox includes an FTP server. You can use this FTP server to transfer files between an xPC TargetBox and a host PC.

After you set up your host PC for network communication, you can connect to an xPC TargetBox using FTP. For information on connecting your host PC to a network, see your system administrator.

**1** On the xPC TargetBox, press the **Reset** button.

**2** You need to cancel the DOS LED test and stop the xPC Target kernel from loading. Press **Ctrl+C** repeatedly until you see the message

```
Terminate batch file 'C:\AUTOEXEC.BAT' (Yes/No/All)?
```

**3** Type **Y**.

The system boots into the work directory, `c:\work`.

**4** Copy the autoexec file configured for running the FTP server into `c:\autoexec.bat`. Type

```
copy c:\autoexec.ftp c:\autoexec.bat
```

**5** Open the FTP configuration file. Type

```
cd c:\work
edit config.tel
```

**6** Change the IP address to the one assigned to your xPC TargetBox. For example, enter

```
myip=192.168.0.1
```

**7** Save and close the file. To do this, press **Alt+F** to open the File menu, use the down arrows to select Exit, and then press **Enter**.

**8** On the xPC TargetBox, press the **Reset** button.

The system boots the FTP server.

```
ftp>
```

## Using FTP Commands

After you start the FTP server on the host PC and passwords are checked, you are prompted for individual FTP commands. These commands are documented in the manuals for the host computer, but most FTP implementations have similar commands because they are modeled after the Berkeley UNIX version of FTP. The following table lists FTP commands that are common to most implementations.

Command	Action
ascii	Set transfer mode to ASCII (default).
binary	Set transfer mode to binary. This is the mode you need to transfer target applications to an xPC TargetBox.
cd <path>	Set a new default directory on an xPC TargetBox.
dir	Show directories and files in an xPC TargetBox default directory.
pwd	Show the current xPC TargetBox directory name.
get <filename>	Get a file from an xPC TargetBox and send it to the host PC.
put <filename>	Send a file from the host PC to an xPC TargetBox. For xPC TargetBox stand-alone applications you need to transfer the files xpcboot.com, autoexec.bat, and <application_name>.rtb.
quit	Exit the FTP server on the host computer.

If you need more information about the File Transfer Protocol, you can find information on the Internet. One Web site is

<http://archive.ncsa.uiuc.edu/SDG/Software/PCTelnet/>

## Copying Files with DOS and FTP

You do not need a floppy disk drive attached to an xPC TargetBox to copy stand-alone applications from the host PC. Instead, you can connect your xPC TargetBox to a network and use FTP to transfer files to the flash disk.

After you create an xPC Target stand-alone application, you can copy the files to the xPC TargetBox flash disk. To create the required stand-alone files, see “StandAlone Mode” on page 4-12.

- 1** Boot the xPC TargetBox with the FTP server. See “Booting xPC TargetBox with FTP Server” on page 4-17.
- 2** On the host computer, open a DOS shell. Start the FTP server with the IP address for your xPC TargetBox. For example, type

```
ftp 192.168.0.1
```

The host PC displays

```
Connected to 192.168.0.1
220 XPC PC FTP server v2.55 -- at 192.168.0.1
User (192.168.0.1:(none)):
```

- 3** Log in to the server by pressing **Enter**.

The host PC logs into the directory `c:\work` and displays the messages

```
230 User logged in
ftp>
```

- 4** Set the host FTP to binary transfer. On the host PC, type

```
binary
```

- 5** Copy files from the host PC to the xPC TargetBox. For example, if you created a stand-alone application from the Simulink model `xpcosc.mdl` in your MATLAB working directory `c:\mwd`, type

```
put c:\mwd\xpcosc_xpc_emb\xpcosc.rtb
```

The host PC displays the following:

```
200 Selected data port
150 Trasferring binary file "xpcosc.com"
226 Transfer complete
ftp: 16536 bytes sent in 0.00 Seconds
ftp>
```

- 6 Copy the boot loader and the autoexec.bat file that loads the kernel and application.

```
put c:\mwd\xpcosc_xpc_emb_xpcboot.com
put c:\mwd\xpcosc_xpc_emb\autoexec.bat
```

- 7 Stop the FTP server. On the xPC TargetBox, type

```
quit
```

The directory changes to c:\xpc.

- 8 Unload the FTP server. Type

```
unldftp
```

- 9 Run the target application. For example, type

```
autoexec
```

This file was created by the build process for a stand-alone application and includes the command `xpcboot xpcosc.rtb`. First the kernel is loaded, then the target application, and finally the target application is started.

- 10 If you want your stand-alone application to run when you press the **Reset** button on your xPC TargetBox,

- a Copy c:\autoexec.bat to a backup file. For example,

```
copy c:\autoexec.bat c:\autobak.bat
```

- b Copy the file c:\autoexec.wrk to c:\autoexec.bat. For example

```
copy c:\autoexec.wrk c:\autoexec.bat
```

Alternatively, edit the file c:\autoexec.bat to include the following lines. Adding these commands to c:\autoexec.bat directs the system to execute the autoexec.bat file located in c:\work.

```
cd c:\work
autoexec
```

---

**Note** Do not confuse `C:\work\autoexec.bat` with `C:\autoexec.bat`. The file `C:\work\autoexec.bat` includes the command `xpcboot.com` to start the xPC Target kernel. The file `C:\autoexec.bat` includes the files you want the system to execute when the system starts up.

---

- 11** On the xPC TargetBox, press the **Reset** button.

## Booting xPC TargetBox to DOS

When you first receive your xPC TargetBox, the system boots the xPC Target kernel and runs a self-test from `c:\tests`. If you want your system to boot into DOS, you need to edit the file `autoexec.bat` or use the preloaded file `autoexec.dos` on the flash disk. xPC TargetBox ships with a copy of FreeDOS preinstalled on the flash disk. Booting your system into DOS enables you to use standard DOS commands such as `edit` and `copy` without also specifying the pathname.

Also, after setting up the kernel (DOSLoader mode) or a stand-alone application (StandAlone mode) to boot from `c:\work`, the xPC TargetBox will always start DOS and then immediately boot the xPC Target kernel. In this case, you might also want to change your system to boot into DOS.

- 1 On the xPC TargetBox, press the **Reset** button.
- 2 When you see the message for loading DOS, press **Ctrl+C** repeatedly until you see the message

```
Terminate batch file 'C:\AUTOEXEC.BAT' (Yes/No/All)?
```

- 3 Type `Y`.

The boot process is stopped and a DOS prompt is displayed. The system boots into either the self-test directory `c:\tests` or the working directory `c:\work`.

- 4 Copy the `autoexec` file configured for booting into DOS to `autoexec.bat`. Type

```
copy c:\autoexec.dos c:\autoexec.bat
```

- 5 On the xPC TargetBox, press the **Reset** button.

The xPC TargetBox boots into DOS and the screen displays

```
Loading FreeDOS
. . .
C:\>
```



Alternatively, you can create a DOS system disk and boot DOS from the floppy disk drive.

---

**Note** On the xPC TargetBox flash disk, FreeDOS is located at `c:\fdos`. If you do not want to boot into DOS, but want to access the FreeDOS commands, specify the pathname, `c:\fdos\bin`, for example `c:\fdos\bin\edit`.

---



# xPC TargetBox I/O Options Reference

---

This chapter describes the I/O board options available with xPC TargetBox. Each option includes one or two test dongles that you attach to the external connectors to run a test. This chapter includes the following sections:

Introduction to I/O Options (p. 5-3)	Describes how to connect your hardware using screw terminal boards and cables included with xPC TargetBox
Loopback Testing of I/O Options (p. 5-5)	Describes how to create your own Simulink models for testing individual I/O options
xPC TargetBox IO 301 (p. 5-10)	Describes Diamond-MM-32-AT (Diamond-MM-32X-AT), an analog input (A/D), analog output (D/A), and digital I/O board
xPC TargetBox IO 302 (p. 5-27)	Describes Diamond Ruby-MM-1612, an analog output (D/A) and digital I/O board
xPC TargetBox IO 303 (p. 5-35)	Describes Diamond Ruby-MM-416, an analog output (D/A) and digital I/O board
xPC TargetBox IO 304 (p. 5-43)	Describes Diamond Onyx-MM, a digital I/O board
xPC TargetBox IO 305 (p. 5-50)	Describes Diamond Quartz-MM-10, a counter and digital I/O board

xPC TargetBox IO 306 (p. 5-64)	Describes Real Time Devices DM6814, an encoder, digital I/O and interrupt board
xPC TargetBox IO 308 (p. 5-74)	Describes Softing CAN-AC2-104, a CAN field bus board
xPC TargetBox IO 309 (p. 5-106)	Describes MPL PATI, a PowerPC controlled analog and timing I/O intelligence board

## Introduction to I/O Options

xPC TargetBox includes screw terminal boards and cables to connect the I/O options to your hardware.

This section includes the following topics:

Pin Layout and Screw Terminal Boards (p. 5-3)	Description of the pin numbers on a screw terminal board that match the pin numbers on an xPC TargetBox connector
Making Your Own Terminal Boards (p. 5-4)	How to use custom hardware connection to add application-specific signal conditioning circuits

---

**Note** The bottom of the xPC TargetBox has a label that describes the configuration of this particular unit. This label indicates the I/O boards in your system, their base addresses, and the ports they are connected to. Replacement configuration labels are available in the lower compartment of the shipping case. Please contact your MathWorks representative if you need more configuration labels.

---

### Pin Layout and Screw Terminal Boards

The pin layout for each I/O option is provided in a table. The numbers printed on each screw terminal board correspond directly to the pin numbers provided in the table.

Each screw terminal board is provided with rubber feet and distance spacers for easy placement on a workbench. A nice feature of the screw terminal boards is that you can stack two or more together to reduce space on your workbench.

## **Making Your Own Terminal Boards**

You might want to make your own connector boards with signal conditioning circuitry specific for your application. The 50-pin connectors on an xPC TargetBox and the screw terminal boards are from the AMPLIMIT .050 series III from Tyco Electronics. The part number is

Tyco Electronics P/N: 787082-5

If you want more information, see <http://www.amp.com> and the TERM50 hardware manual included in the xPC TargetBox shipping case.

## Loopback Testing of I/O Options

Use a test to initially determine if your xPC TargetBox is working correctly, determine if your xPC TargetBox is continuing to work correctly, and to learn more about an I/O option. This section includes the following topics:

Loopback Testing Process (p. 5-5)	How to connect a test dongle to an xPC TargetBox connector and run a target application specific for that connector and I/O option
Determining the Success of a Loopback Test (p. 5-6)	How to observe a scope on the xPC TargetBox monitor that displays the results from a test
Uses for Loopback Testing (p. 5-6)	How to test your xPC TargetBox when you first unpack the box and during use, and how to learn about an I/O option
Creating a Simulink Model for Loopback Testing (p. 5-7)	How to recreate the Simulink model to test all or some of the I/O options in your xPC TargetBox
Running the Target Application for Specific I/O Option Testing (p. 5-8)	How to create and run a target application to test I/O options

### Loopback Testing Process

Loopback testing is a technique to make sure that an xPC TargetBox is working properly with a certain I/O option or all I/O options working together. The technique consists of

- 1 A simple Simulink model with appropriate I/O driver blocks and analysis blocks.
- 2 A target application built from a Simulink model using Real-Time Workshop, xPC Target, and a C/C++ compiler.
- 3 Running the target application on the xPC TargetBox where the corresponding I/O option is exercised. Signals generated at the outputs

are looped back through physical connectors to the inputs of the same I/O option.

- 4** The analysis part of the target application compares the generated signal with the looped-back signal. Depending on the result of the comparison, a scope displays success = 1 or failure = 0.

## Determining the Success of a Loopback Test

A successful test for an I/O option ensures that the following are functioning properly:

- xPC TargetBox is running properly, and it can execute a target application.
- The I/O option (I/O board) is accessed through its base address. This address was set by the manufacturer when you purchased your xPC TargetBox. This is the address you enter in the `xpctgboxtest` command to create a Simulink model for testing.
- The I/O option, consisting of the I/O board, I/O connector, test dongle, I/O cable, and screw terminal board, is fully functional both mechanically and electrically.

A failed test of an I/O option signifies that one of these parts has a failure that makes additional testing necessary.

## Uses for Loopback Testing

You can use testing of an I/O option to accomplish various testing tasks at different stages when using an xPC TargetBox. Some uses of testing are

- Initial full system test of the xPC TargetBox and all I/O options when unpacking the system for the first time after delivery

At the time you unpack an xPC TargetBox, you can attach test dongles and a monitor, power up the system, and immediately observe the proper functioning of the xPC TargetBox and all installed I/O options.

- Testing a particular I/O option during xPC TargetBox operation

This task is useful if you are working with xPC Target and the xPC TargetBox, and you believe that something might be wrong with an I/O option. For example, you could have a damaged I/O board. This may also



be useful when you are working with a more complex real-world model (application) and it is not working to your satisfaction. You can then use testing of a particular I/O option as a unit test for a certain feature of your xPC TargetBox to narrow down the possible causes of the overall failure.

- Getting familiar with a certain I/O option and its features

## Creating a Simulink Model for Loopback Testing

During the final system test at the manufacturer, a Simulink model was created that does a test of all I/O options installed on your system. This Simulink model was built into a stand-alone target application and put onto the flash disk of your xPC TargetBox. The stand-alone application is called `xpctgtst.rtb` and is located in the directory `c:\work`.

It is good practice to back up this file for later use in case the flash disk is corrupted. Nevertheless, you can create the same Simulink model on which `xpctgtst.rtb` is based at any time. You can then build and download that target application and redo the entire system test if the original stand-alone test application no longer exists:

- 1 On the bottom of the xPC TargetBox, check the configuration label for the I/O options installed in your xPC TargetBox and the base addresses.

Find the jumper codes for your I/O options. Use the table in “Configuration Label for I/O Options” on page 2-19 to determine the jumper settings from the jumper codes.

- 2 Enter the command to create the I/O option Simulink model. For example, if you want a model to test I/O option 301 and I/O option 305 with base addresses 0x300 and 0x280, in the MATLAB Command Window on the host PC, type

```
xpctgboxtest({'I0301', '0x300', 'C21'}, {'I0305', '0x280'})
```

A Simulink test model is created. This model is the system test model that was used to create the target application initially delivered on the flash disk as a stand-alone application. You can use this model to build and run a target application on the xPC TargetBox.

- 3 Save the Simulink test model in a directory outside the MATLAB root directory.

---

**Caution** Make sure that you have connected the correct dongles to the correct I/O connectors; otherwise, you can damage an I/O option or even the entire xPC TargetBox.

---

## Running the Target Application for Specific I/O Option Testing

If you want to exercise a test of a certain I/O option, use the following steps:

- 1** Power down the xPC TargetBox. Never plug or unplug any connectors while the xPC TargetBox is powered up.
- 2** Select the correct test dongle or dongles. Some I/O options come with two dongles. Each test dongle is labeled with the I/O option number (for example, IO 301) and its port number. For example, IO 301 has two ports labeled IO 301-1 (analog) and IO 301-2 (digital).

On the bottom panel of the xPC TargetBox, check the xPC TargetBox configuration label. This label lists the I/O connector to connect the test dongle to. Also write down the base address for the I/O option (I/O board) you want to exercise.

---

**Caution** Make sure that you connect the selected test dongles to the corresponding I/O connectors; otherwise, you can damage an I/O option or the entire xPC TargetBox when powering it up again.

---

- 3** Boot the xPC TargetBox so that the xPC Target kernel starts running using either
  - BootFloppy mode from a floppy disk.
  - DOSLoader mode from the flash disk.
- 4** From the MATLAB Command Window, type

```
xpctargetping
```

This command checks for a proper connection between your xPC TargetBox and host PC, and if successful, responds with

```
ans =  
success
```

- 5 Enter the command to create the I/O option Simulink model. For example, if you want a model to exercise I/O option 301 with the I/O board base address 0x300, with the analog input jumper set for differential mode, and the analog output jumper set to  $\pm 10$  volts, type

```
xpctgboxtest({'I0301','0x300','C21'})
```

The corresponding Simulink test model is created.

- 6 Save the Simulink model.
- 7 In the Simulink window, and from the **Tools** menu, point to **Real-Time Workshop**, and then click **Build**.

Real-Time Workshop, xPC Target, and a C/C++ compiler build and download a target application to the xPC TargetBox.

- 8 If the build and download process was successful, type

```
+tg
```

The target application starts running.

- 9 Check the numerical values on the scopes.
  - If the scopes show numerical values of 1, then the test was successful and the problem with your application might be somewhere else.
  - If one of the displayed values is 0, then either the board or the test dongle might be damaged. In this case, contact your MathWorks representative for help.

## xPC TargetBox IO 301

The Diamond-MM-32-AT is an analog and digital board with 32 (16-bit) analog input (A/D) channels, four (12-bit) analog output (D/A) channels, and 24 digital I/O lines. Your xPC TargetBox IO 301 option might contain the Diamond-MM-32X-AT board instead of the Diamond-MM-32-AT. The information in this chapter applies to both boards.

This section includes the following topics:

- “Wiring for IO 301 Test Dongles” on page 5-10
- “Pin Numbering for Connectors and Screw Terminal Boards” on page 5-11
- “Pin Layout IO 301 (1 of 2 Connectors)” on page 5-11
- “Pin Layout IO 301 (2 of 2 Connectors)” on page 5-13
- “Testing Model IO 301” on page 5-14
- “Diamond-MM-32-AT (IO 301) Driver Blocks” on page 5-16

---

**Note** xPC TargetBox does not support the counters on this board.

---

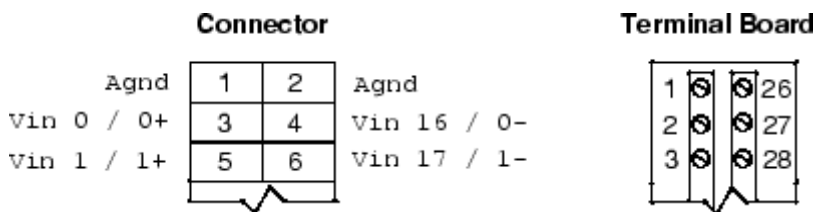
### Wiring for IO 301 Test Dongles

xPC TargetBox communicates with this board using two connectors. Some of the pins in the test dongles are connected together to allow for testing of this board:

- Analog Header (J3) dongle — The following pins are connected together: 3-38, 7-37, 11-35, 15-36, and 2-4-8-12-16.
- Digital Header (J4) dongle — The following pins are connected together: 12-20 and 16-24.

## Pin Numbering for Connectors and Screw Terminal Boards

The terminals on the screw terminal boards are numbered sequentially, but the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the connector figures.



### Pin Layout IO 301 (1 of 2 Connectors)

The following figure is the first pin layout for the IO 301 option connector. This is the J3 connector in the manufacturer's data sheet at <http://www.diamondsystems.com/files/binaries/DMM32v2.64.pdf> ([http://www.diamondsystems.com/files/binaries/DMM32XAT\\_v1.00.pdf](http://www.diamondsystems.com/files/binaries/DMM32XAT_v1.00.pdf) for Diamond-MM-32X-AT).

The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.

	Agnd	1	2	Agnd	
Analog input	Vin 0 / 0+	3	4	Vin 16 / 0-	Analog input
	Vin 1 / 1+	5	6	Vin 17 / 1-	
	Vin 2 / 2+	7	8	Vin 18 / 2-	
	Vin 3 / 3+	9	10	Vin 19 / 3-	
	Vin 4 / 4+	11	12	Vin 20 / 4-	
	Vin 5 / 5+	13	14	Vin 21 / 5-	
	Vin 6 / 6+	15	16	Vin 22 / 6-	
	Vin 7 / 7+	17	18	Vin 23 / 7-	
	Vin 8 / 8+	19	20	Vin 24 / 8-	
	Vin 9 / 9+	21	22	Vin 25 / 9-	
	Vin 10 / 10+	23	24	Vin 26 / 10-	
	Vin 11 / 11+	25	26	Vin 27 / 11-	
	Vin 12 / 12+	27	28	Vin 28 / 12-	
	Vin 13 / 13+	29	30	Vin 29 / 13-	
	Vin 14 / 14+	31	32	Vin 30 / 14-	
	Vin 15 / 15+	33	34	Vin 31 / 15-	
Analog output	Vout 3	35	36	Vout 2	Analog output
	Vout 1	37	38	Vout 0	
	Vref Out	39	40	Agnd	
	A/D Convert	41	42	Ctr 2 Out / Dout 2	
	Dout 1	43	44	Ctr 0 Out / Dout 0	
	Extclk /Din 3	45	46	Extgate / Din 2	
	Gate 0 / Din 1	47	48	Clk 0 / Din 0	
	+5V	49	50	Dgnd	

## Pin Layout IO 301 (2 of 2 Connectors)

The following figure is the second pin layout for the IO 301 option connector. This is the J4 connector in the manufacturer's data sheet at <http://www.diamondsystems.com/files/binaries/DMM32v2.64.pdf> ([http://www.diamondsystems.com/files/binaries/DMM32XAT\\_v1.00.pdf](http://www.diamondsystems.com/files/binaries/DMM32XAT_v1.00.pdf) for Diamond-MM-32X-AT).

The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.

Digital I/O Port A	A7	1	2	A6	Digital I/O Port A
	A5	3	4	A4	
	A3	5	6	A2	
	A1	7	8	A0	
Digital I/O Port B	B7	9	10	B6	Digital I/O Port B
	B5	11	12	B4	
	B3	13	14	B2	
	B1	15	16	B0	
Digital I/O Port C	C7	17	18	C6	Digital I/O Port C
	C5	19	20	C4	
	C3	21	22	C2	
	C1	23	24	C0	
	Latch	25	26	Ack	
	NC	27	28	NC	
	NC	29	30	NC	
	NC	31	32	NC	
	+5V	33	34	Dgnd	

---

**Note** This connector does not use pins 35 to 50.

---

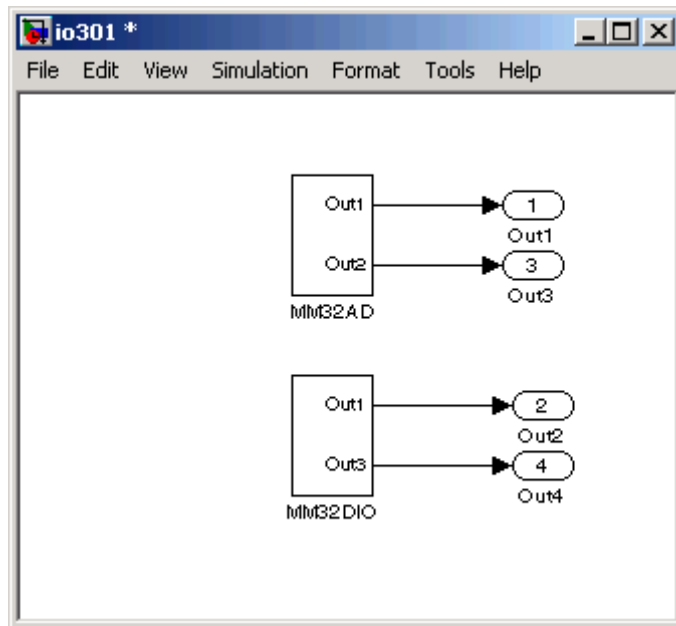
## Testing Model IO 301

This model tests the analog channels. A constant value is written to an A/D driver block and then read from the D/A driver block. The values are compared with an error defined in a second constant block:

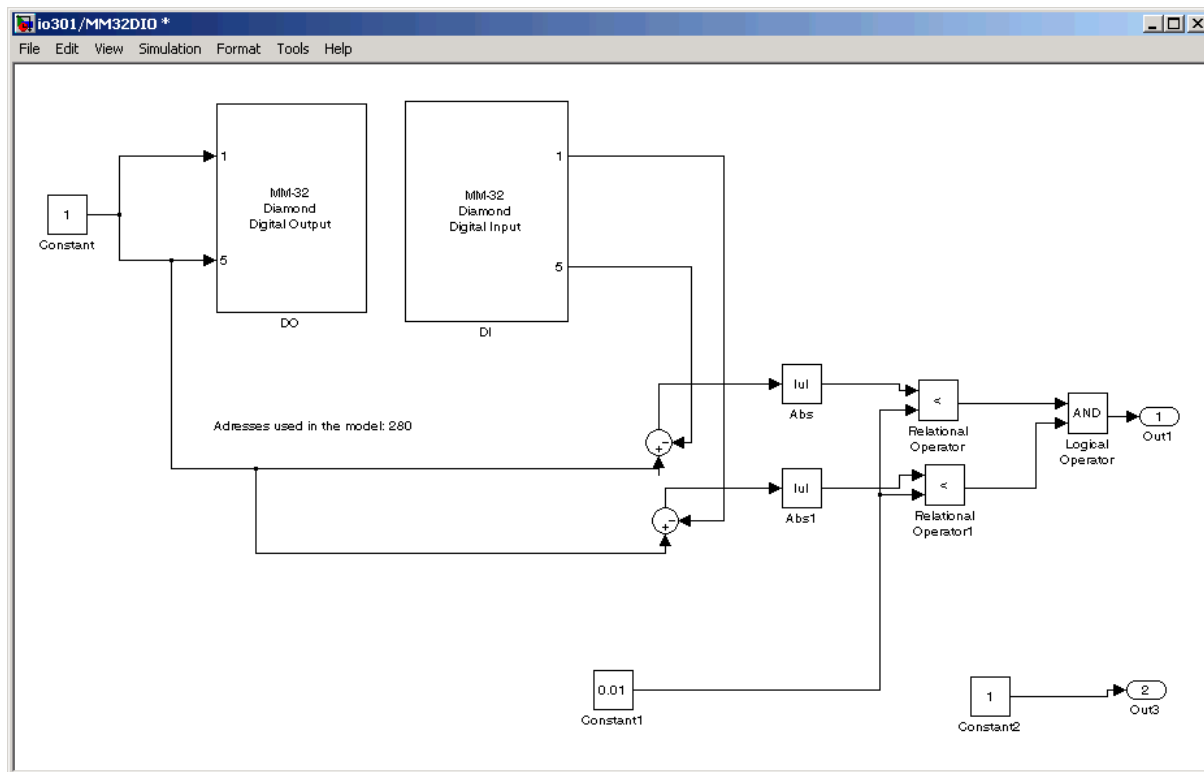
- 1 In the MATLAB Command Window, type

```
io301
```

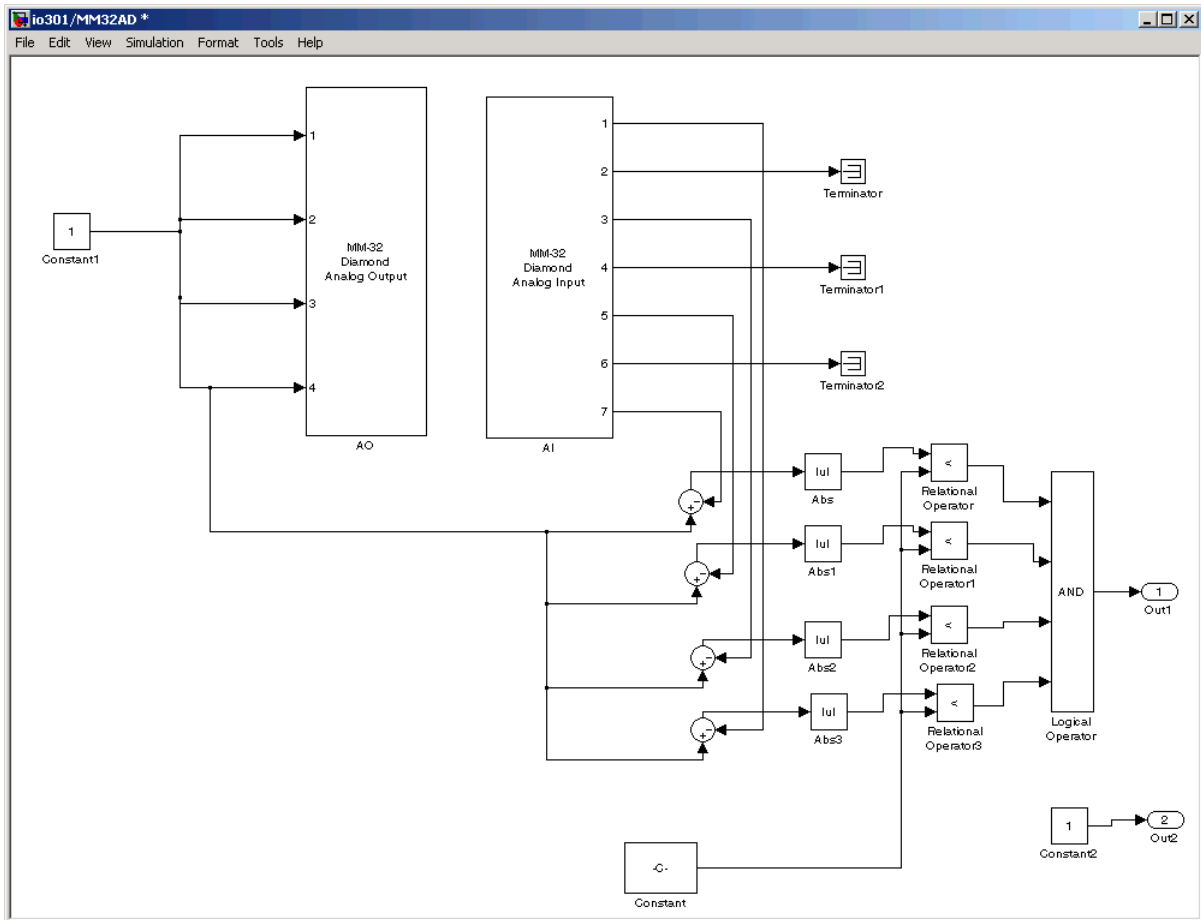
The Simulink model for testing the IO 301 option opens.







2 Double-click the MM32AD block to see the associated model.



**Diamond-MM-32-AT (IO 301) Driver Blocks**

The Diamond-MM-32-AT is a PC/104 I/O board with 32 single or 16 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 200 kHz, four analog output (D/A) channels (12-bit), and 24 digital input and output lines.

xPC TargetBox supports this board with these driver blocks. These blocks also support the Diamond-MM-32X-AT board.

- “Diamond-MM-32-AT Analog Input (A/D, IO 301)” on page 5-17
- “Diamond-MM-32-AT Frame Analog Input (A/D, IO 301)” on page 5-18
- “Diamond-MM-32-AT Analog Output (D/A, IO 301)” on page 5-22
- “Diamond-MM-32-AT Digital Input (IO 301)” on page 5-23
- “Diamond-MM-32-AT Digital Output (IO 301)” on page 5-24

### Board Characteristics

Board name	Diamond-MM-32-AT
Manufacturer	Diamond Systems Corporation
Bus type	PC/104
Access method	I/O Mapped
Multiple block instance support	A/D: No D/A: Yes DIO: Yes
Multiple board support	Yes

### Diamond-MM-32-AT Analog Input (A/D, IO 301)

#### Scaling Input to Output.

Hardware Input	Block Output Data Type	Scaling
Volts	double	1

#### Driver Block Parameters.

**Channel configuration** — From the list, select the following. Refer to the Diamond-MM-32-AT documentation for a description of the configuration modes:

- 1-32 SE to select the configuration mode labeled A (32 single-ended input channels)
- 1-16 DI to select the configuration mode labeled B (16 differential input channels)
- 1-8SE 9-16 DI 17-24 SE to select the configuration mode labeled D (eight single-ended input channels labeled 1 through 8, eight differential input channels labeled 9 through 16, and an additional eight single-ended input channels labeled 17 through 24).

Note that the selected channel configuration must match the configuration set by the jumpers in block J5. This driver does not support mode C.

**First channel number** — Enter the number of the first channel in a set of contiguous channels. Depending on the channel configuration selected, the first channel number must be within the range 1 through 32, 1 through 16, or 1 through 24.

**Number of channels** — Enter the number of input channels you want to use. The maximum number of channels varies between 1 and 32 and depends on **Channel configuration** and the **First channel number**.

**Range** — From the list, choose a voltage range. The input range applies to all channels.

**Sample time** — Enter the base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

### **Diamond-MM-32-AT Frame Analog Input (A/D, IO 301)**

The Diamond-MM-32-AT Frame Analog Input block is frame-based. A frame consists of a fixed number of samples (defined by the **Number of scans per frame** parameter) for each of a specified set of channels. A scan is a group of samples, one for each channel.

Normally, the system timer controls an xPC Target model at intervals specified by the block **Sample Time** parameter. In contrast, the Diamond-MM-32-AT Frame Analog Input block executes the model in which it occurs each time it converts a new frame of data. You control this rate with the parameter values **Interval between scans** and **Number of scans per frame**:

$$\text{Rate} = (\text{Interval between scans}) \times (\text{Number of scans per frame})$$

You control the frame size with the parameter values **Number of channels** and **Number of scans per frame**:

$$\text{frameSize} = (\text{Number of channels}) \times (\text{Number of scans per frame})$$

After the block assembles a frame of data, it generates an interrupt, which triggers the next iteration of the model.

After you add this block to a model and are ready to configure the model, edit the xPC Target code generation options:

- 1** From the model, select **Simulation > Configuration Parameters**.
- 2** Select **Real-Time Workshop**.
- 3** In the **General** tab, from the **RTW system target file** list, browse to and select `xpctarget.tlc`.
- 4** In the **xPC Target code generation options tab**, from the **I/O board generating the interrupt** list, select the value `Diamond-MM-32`. This specifies that the Diamond-MM-32-AT board generates the interrupt.
- 5** In the same tab, from the **Real-time interrupt source** list, select the IRQ number you have jumpered on the board.
- 6** In the same tab, for the **PCI slot/ISA base address** parameter, enter the same ISA address as for the Diamond-MM-32-AT Frame block **Base address** parameter.
- 7** Select **OK** and save the model.

**Note** To enable the interrupt for the frame-based block, the 16-bit bus jumper must be present in the board J7 jumper block. For further details, see the section labeled “16-Bit Bus” in the Diamond-MM-32-AT user’s guide documentation (<http://www.diamondsystems.com/files/binaries/DMM32v2.64.pdf>). Alternatively, see “16-Bit Data Bus” in the Diamond-MM-32X-AT user’s guide documentation ([http://www.diamondsystems.com/files/binaries/DMM32XAT\\_v1.00.pdf](http://www.diamondsystems.com/files/binaries/DMM32XAT_v1.00.pdf)).

**Scaling Input to Output.**

Hardware Input	Block Output Data Type	Scaling
Volts	double	1

**Driver Block Parameters.**

**Channel configuration** — From the list, select the following. Refer to the Diamond-MM-32-AT documentation for a description of the configuration modes.

- 1-32 Single-Ended to select the configuration mode labeled A (32 single-ended input channels)
- 1-16 Differential to select the configuration mode labeled B (16 differential input channels)
- 1-8 and 7-24 Single-Ended; 9-16 Differential to select the configuration mode labeled D (eight single-ended input channels labeled 1 through 8, eight differential input channels labeled 9 through 16, and an additional eight single-ended input channels labeled 17 through 24).

Note that the selected channel configuration must match the configuration set by the jumpers in block J5. This driver does not support mode C.

**Output Signal Type** — From the list, select either Vector or Frame.

- **Vector** — Select Vector if you expect the output signal from this block to be the input signal for an xPC Target scope or some other block that requires vector input.
- **Frame** — Select Frame if you expect the output signal from this block to be the input signal for a block that requires a frame, for example, a Signal Processing block.

**Range** — From the list, select a voltage range. The input range applies to all channels.

**First channel number** — Enter the number of the first channel in a set of contiguous channels. Depending on the value of the **Channel Configuration** parameter, the first channel number must be within the range 1 through 32, 1 through 16, or 1 through 24.

Number the channels starting from 1 even if Diamond Systems numbers them starting from 0.

**Number of channels** — Enter the number of input channels you want to use. The maximum number of channels varies between 1 and 32 and depends on **Channel configuration** and the **First channel number**. Note that hardware limitations require that either the **Number of channels** or **Number of scans per frame** value be even. If your application requires that both quantities be odd, add 1 to one of them and ignore the resulting additional data.

**Number of scans per frame** — Enter the number of scans per frame. For a value of N, each output port of the block will have a signal width of N and contain N samples of the corresponding channel. Note that hardware limitations require that either the **Number of channels** or **Number of scans per frame** value be even. If your application requires that both quantities be odd, add 1 to one of them and ignore the resulting additional data.

**Interval between conversions within a scan** — From the list, select the interval, in microseconds, between conversions within a scan.

**Interval between scans** — Enter the interval, in seconds, between successive scans.

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

### **Diamond-MM-32-AT Analog Output (D/A, IO 301)**

#### **Scaling Input to Output.**

<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
Volts	double	1

#### **Driver Block Parameters.**

**Channel vector** — Enter numbers between 1 and 4. This driver allows the selection of individual D/A channels in any order. The number of elements defines the number of D/A channels you use. For example, to use the first and second analog output (D/A) channels, enter

[1,2]

Number the channels beginning with 1 even if the board manufacturer starts numbering the channels with 0.

**Range** — From the list, choose a range code. This driver does not allow a different range for each of the four channels. This selection must correspond to the range and bipolar/unipolar jumper settings on the board.

The following table is a list of the ranges for this driver and the corresponding range codes. The D/A specific jumpers on the board must be in the correct positions for the ranges entered.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5



Input Range (V)	Range Code
0 to +10	10
0 to +5	5

For example, if the first and second channel range is 0 to +10 volts, enter

[10, 10]

**Reset vector** — The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is used for all channels. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 0, the channel remains at the last value attained while the model was running.

**Initial value vector** — The initial value vector contains the initial voltage values for the output channels. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that value is the initial value for all channels. The channels are set to the initial values between the time the model is downloaded and the time it is started.

**Sample time** — Enter the base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

### **Diamond-MM-32-AT Digital Input (IO 301)**

The Diamond-MM-32-AT has one 8255 chip with three ports (A,B,C). Each port has a maximum of eight digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital input driver block, you configure the port input.

**Scaling Input to Output.**

Hardware Input	Block Output Data Type	Scaling
TTL	double	TTL low = 0.0 TTL high = 1.0

**Driver Block Parameters.**

**Channel vector** — Enter numbers between 1 and 8 to select the digital input lines used with this port. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital inputs for one port, enter

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

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — From the list choose A, B, or C. The I/O board has an 8255 chip with three ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of eight digital lines that you can configure as inputs or outputs depending on which driver block is chosen. In each case, one block is needed for each port.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

**Diamond-MM-32-AT Digital Output (IO 301)**

The Diamond-MM-32-AT has one 8255 chip with three ports (A,B,C). Each port has a maximum of eight digital I/O lines that you can configure as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, you can configure the port as output.

### Scaling Input to Output.

Hardware Output	Block Input Data Type	Scaling
TTL	double	< 0.5 = TTL low > 0.5 = TTL high

### Driver Block Parameters.

**Channel vector** — Enter numbers between 1 and 8 to select the digital output lines used with this port. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital outputs for one port, enter

```
[1,2,3,4,5,6,7,8]
```

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — From the list choose either A, B, or C. The I/O board has an 8255 chip with three ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of eight digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is needed for each port.

**Reset vector** — The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is used for all channels. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 0, the channel remains at the last value attained while the model was running.

**Initial value vector** — The initial value vector contains the initial values (0 or 1) of the output channels. Enter a scalar or a vector that is the same length as the channel vector. If you enter a scalar, that value is used for all channels.

The channels are set to these initial values between the time the model is downloaded and the time it is started.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

## xPC TargetBox IO 302

The Diamond Ruby-MM-1612 is an analog and digital board with 16 (12-bit) analog output (D/A) channels and 24 digital I/O lines. This section includes the following topics:

- “Wiring for IO 302 Test Dongle” on page 5-27
- “Pin Layout IO 302” on page 5-27
- “Testing Model IO 302” on page 5-29
- “Ruby-MM-1612 (IO 302) Driver Blocks” on page 5-30

---

**Note** xPC TargetBox does not support the external trigger on this board.

---

### Wiring for IO 302 Test Dongle

xPC TargetBox communicates with this board using one connector. Some of the pins in the test dongle are connected together to allow for testing of this board.

Header (J3) Dongle — The following pins are connected together: 4-46 and 8-44.

### Pin Layout IO 302

The following figure is the pin layout for the IO 302 option connector. This is the J3 connector in the manufacturer’s data sheet at <http://www.diamondsystems.com/files/binaries/RMM1612v11.pdf>.

The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.

	Agnd	1	2	Vout 0	} Analog output	
	Agnd	3	4	Vout 1		
	Agnd	5	6	Vout 2		
	Agnd	7	8	Vout 3		
	Agnd	9	10	Vout 4		
	Agnd	11	12	Vout 5		
	Agnd	13	14	Vout 6		
	Agnd	15	16	Vout 7		
} Analog output	Vout 8	17	18	Vout 9		
	Vout 10	19	20	Vout 11		
	Vout 12	21	22	Vout 13		
	Vout 14	23	24	Vout 15		
} Digital I/O Port A	DIO A7	25	26	DIO A6		} Digital I/O Port A
	DIO A5	27	28	DIO A4		
	DIO A3	29	30	DIO A2		
	DIO A1	31	32	DIO A0		
} Digital I/O Port B	DIO B7	33	34	DIO B6	} Digital I/O Port B	
	DIO B5	35	36	DIO B4		
	DIO B3	37	38	DIO B2		
	DIO B1	39	40	DIO B0		
} Digital I/O Port C	DIO C7	41	42	DIO C6	} Digital I/O Port C	
	DIO C5	43	44	DIO C4		
	DIO C3	45	46	DIO C2		
	DIO C1	47	48	DIO C0		} Ext Trig
+5V	49	50	Dgnd			

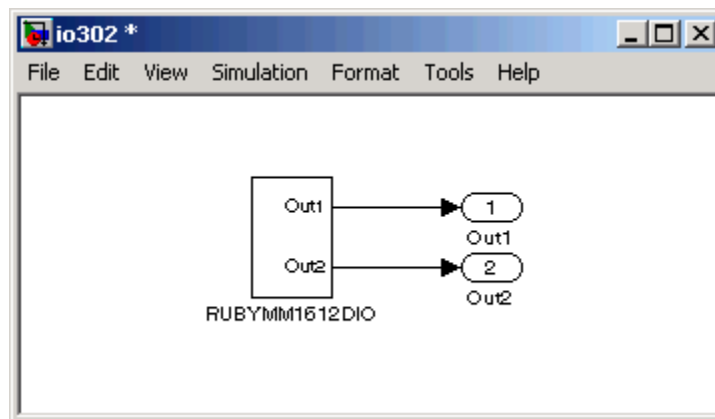
## Testing Model IO 302

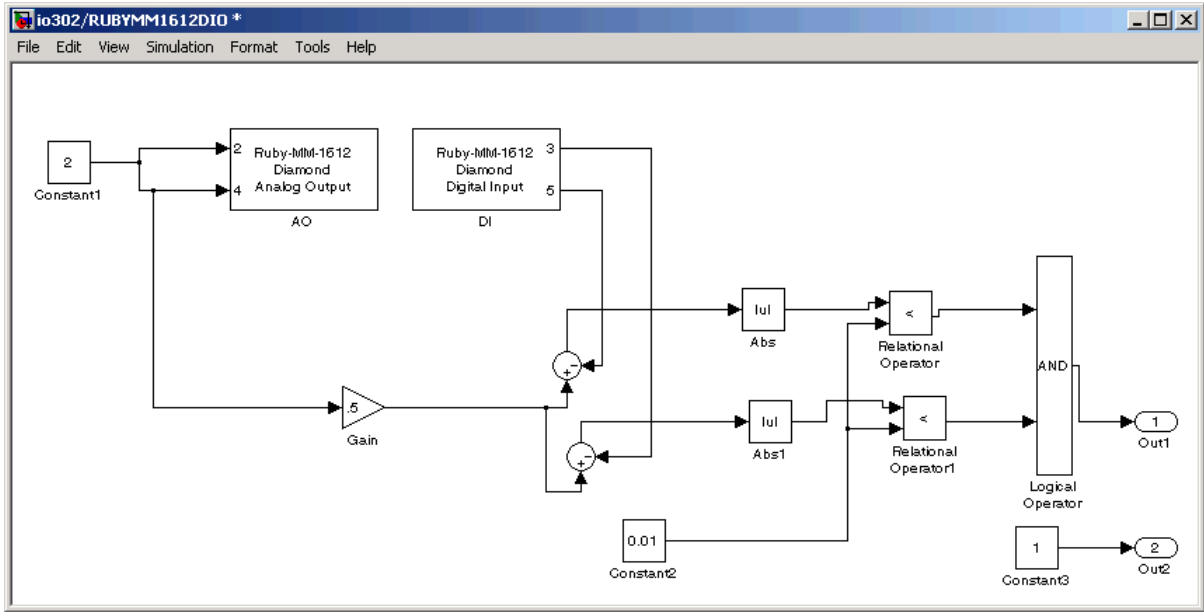
A constant value is written to the A/D driver block and then read from the D/A driver block. The values are compared with an error defined in a second constant block, and the results sent to an Output block:

- 1 In the MATLAB Command Window, type

```
io302
```

The Simulink model for testing the IO 302 option opens.





## Ruby-MM-1612 (IO 302) Driver Blocks

The Ruby-MM-1612 is an I/O board with 16 (12-bit) analog output (D/A) channels and 24 digital I/O lines that can be configured in groups of eight for either input or output.

xPC TargetBox supports this board with these driver blocks:

- “Ruby-MM-1612 Analog Output (D/A, IO 302)” on page 5-31
- “Ruby-MM-1612 Digital Input (IO 302)” on page 5-32
- “Ruby-MM-1612 Digital Output (IO 302)” on page 5-33

### Board Characteristics

Board name	Ruby-MM-1612
Manufacturer	Diamond Systems Corporation
Bus type	PC/104



## Board Characteristics (Continued)

Access method	I/O Mapped
Multiple block instance support	D/A:No DIO:Yes
Multiple board support	Yes

## Ruby-MM-1612 Analog Output (D/A, IO 302)

### Scaling Input to Output.

Hardware Output	Block Input Data Type	Scaling
Volts	double	1

### Driver Block Parameters.

**Channel vector** — Enter a vector containing channel numbers between 1 and 16. This driver allows the selection of individual D/A channels in any order. The number of elements defines the number of D/A channels used. For example, to use the first and second analog output (D/A) channels, enter

[1,2]

Number the channels beginning with 1 even if the board manufacturer starts numbering the channels with 0.

**Range for bank 1, Range for bank 2** — Bank 1 consists of channels 1 to 8 and bank 2 consists of channels 9 to 16. The output range can be specified on a per-bank basis. These ranges must correspond to the jumper settings in header R4 on the board. See the board manual for details.

Note that if you select a range of either -5V to +5V, 0 to +5V, -2.5V to +2.5V, or 0 to +2.5V for one bank, then it is not possible to select a range of either -10V to +10V or 0 to +10V for the other bank. This is because jumper 5 in header J4 (On-Board Reference Full-Scale Voltage Selection) affects all channels, not just those of a single bank. See the board manual for details.

This driver supports the Adjustable Reference Voltage. You can use this feature with either output range -2.5V to +2.5V or 0 to +2.5V. If for example you adjust potentiometer R4 to 2.3 V (instead of the default setting of 2.5), then an input signal of 1.2 results in an output voltage of  $(1.2 / 2.5) * 2.3 \text{ V} = 1.1 \text{ V}$ . See the board manual for details.

**Reset vector** — The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is used for all channels. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 0, the channel remains at the last value attained while the model was running.

**Initial value vector** — The initial value vector contains the initial voltage values for the output channels. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that value is the initial value for all channels. The channels are set to the initial values between the time the model is downloaded and the time it is started. If you provide a value that is out of the channel's range, the value is reset to the lower or upper range value.

**Sample time** — Base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

### **Ruby-MM-1612 Digital Input (IO 302)**

Ruby-MM-1612 boards have three I/O ports, each containing eight digital I/O lines. These ports can be configured independently for either input or output. Use a separate driver block for each port. By selecting the digital input driver block for a given port, that port is configured for input.

### Scaling Input to Output.

Hardware Input	Block Output Data Type	Scaling
TTL	double	TTL low = 0.0 TTL high = 1.0

### Driver Block Parameters.

**Channel vector** — Enter numbers between 1 and 8 to select the digital input lines used with this port. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital inputs for the current port, enter

```
[ 1, 2, 3, 4, 5, 6, 7, 8 ]
```

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — From the list choose A, B, or C.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the base address board setting. For example, if the base address is 300 (hexadecimal), enter

```
0x300
```

### Ruby-MM-1612 Digital Output (IO 302)

Ruby-MM-1612 series boards have three I/O ports, each containing eight digital I/O lines. These ports can be configured independently for either input or output. Use a separate driver block for each port. By selecting the digital output driver block for a given port, you configure that port for output.

**Scaling Input to Output.**

Hardware Input	Block Output Data Type	Scaling
TTL	double	TTL low = 0.0 TTL high = 1.0

**Driver Block Parameters.**

**Channel vector** — Enter numbers between 1 and 8 to select the digital output lines used with this port. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital outputs for the current port, enter

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

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — From the list choose A, B, or C.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the base address board setting. For example, if the base address is 300 (hexadecimal), enter

0x300

## xPC TargetBox IO 303

The Diamond Ruby-MM-416 is an analog and digital board with four (16-bit) analog output (D/A) channels and 24 digital I/O lines. This section includes the following topics:

- “Wiring for IO 303 Test Dongle” on page 5-35
- “Pin Layout IO 303” on page 5-35
- “Testing Model IO 303” on page 5-37
- “Ruby-MM-416 (IO 303) Driver Blocks” on page 5-38

---

**Note** xPC TargetBox does not support the external trigger on this board.

---

### Wiring for IO 303 Test Dongle

xPC TargetBox communicates with this board using one connector. Some of the pins in the test dongle are connected together to allow for testing of this board.

Header (J3) dongle — The following pins are connected together: 4-46 and 8-44.

### Pin Layout IO 303

The following figure is the pin layout for the IO 303 option connector. This is the J3 connector in the manufacturer’s data sheet at <http://www.diamondsystems.com/files/binaries/RMM416v11.pdf>.

The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.

	Agnd	1	2	Vout 0	} Analog output
	Agnd	3	4	Vout 1	
	Agnd	5	6	Vout 2	
	Agnd	7	8	Vout 3	
	NC	9	10	NC	
	NC	11	12	NC	
	NC	13	14	NC	
	NC	15	16	NC	
	NC	17	18	NC	
	Agnd	19	20	+15V	
	-15V	21	22	Agnd	
	Dgnd	23	24	Ext Trig	
Digital I/O Port A	DIO A7	25	26	DIO A6	} Digital I/O Port A
	DIO A5	27	28	DIO A4	
	DIO A3	29	30	DIO A2	
	DIO A1	31	32	DIO A0	
Digital I/O Port C	DIO C7	33	34	DIO C6	} Digital I/O Port C
	DIO C5	35	36	DIO C4	
	DIO C3	37	38	DIO C2	
	DIO C1	39	40	DIO C0	
Digital I/O Port B	DIO B7	41	42	DIO B6	} Digital I/O Port B
	DIO B5	43	44	DIO B4	
	DIO B3	45	46	DIO B2	
	DIO B1	47	48	DIO B0	
	+5V	49	50	Dgnd	

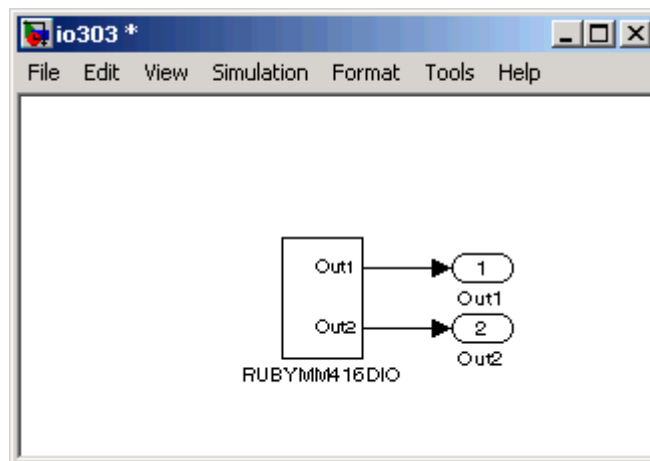
## Testing Model IO 303

A constant value is written to the A/D driver block and then read from the digital input driver block. The values are compared with an error defined in a second constant block, and the results sent to an Output block:

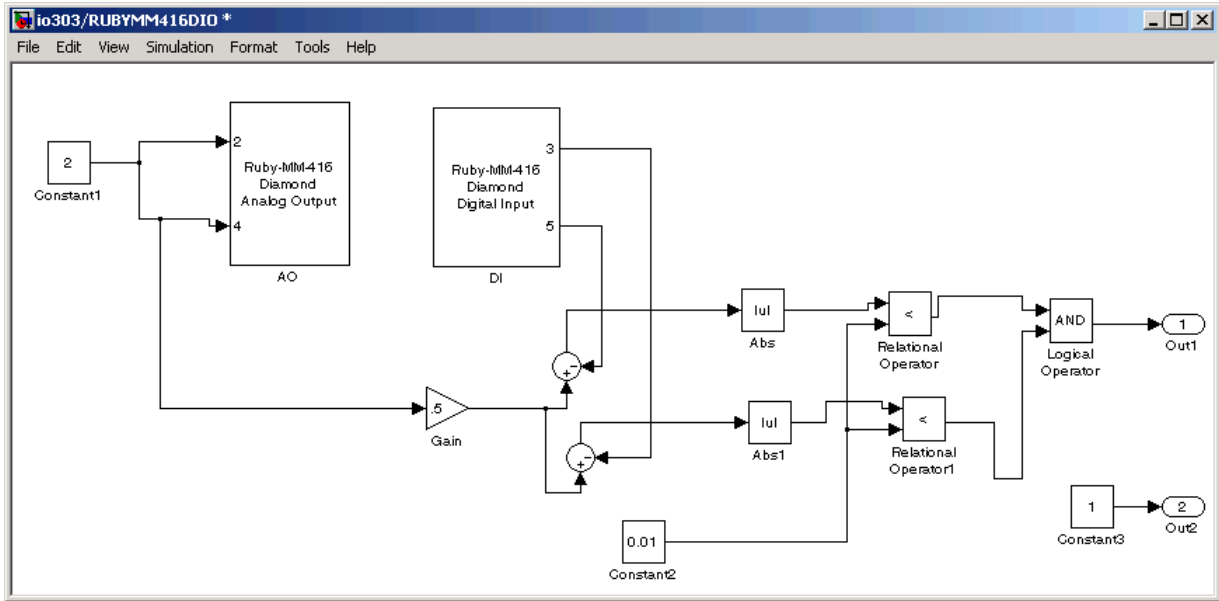
- 1 In the MATLAB Command Window, type

```
io303
```

The Simulink model for testing the IO 303 option opens.



You can double-click the RUBYMM416DIO block to bring up the model.



## Ruby-MM-416 (IO 303) Driver Blocks

The Ruby-MM-416 is an I/O board with four 16-bit analog output (D/A) channels and 24 digital I/O lines that can be configured in groups of eight for either input or output.

xPC TargetBox supports this board with these driver blocks:

- “Ruby-MM-416 Analog Output (D/A, IO 303)” on page 5-39
- “Ruby-MM-416 Digital Input (IO 303)” on page 5-40
- “Ruby-MM-416 Digital Output (IO 303)” on page 5-41

## Board Characteristics

Board name	Ruby-MM-416
Manufacturer	Diamond Systems Corporation
Bus type	PC/104
Access method	I/O Mapped



### Board Characteristics (Continued)

Multiple block instance support	D/A:No DIO:Yes
Multiple board support	Yes

### Ruby-MM-416 Analog Output (D/A, IO 303)

#### Scaling Input to Output.

Hardware Output	Block Input Data Type	Scaling
Volts	double	1

#### Driver Block Parameters.

**Channel vector** — Enter a vector containing channel numbers between 1 and 4. This driver allows the selection of individual D/A channels in any order. The number of elements defines the number of D/A channels used. For example, to use the first and second analog output (D/A) channels, enter

[1,2]

Number the channels beginning with 1 even if the board manufacturer starts numbering the channels with 0.

**Range vector** — The range vector must be a scalar or a vector the same length as the channel vector. The vector entries must use range codes from the following table:

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
0 to 10	10

The range codes you enter must be consistent with the jumper settings on the board.

**Reset vector** — The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is used for all channels. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 0, the channel remains at the last value attained while the model was running.

**Initial value vector** — The initial value vector contains the initial voltage values for the output channels. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that value is the initial value for all channels. The channels are set to the initial values between the time the model is downloaded and the time it is started. If you provide a value that is out of the channel’s range, the value is reset to the lower or upper range value.

**Sample time** — Base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the base address setting on the board (header J6). For example, if the base address is 300 (hexadecimal), enter

0x300

### **Ruby-MM-416 Digital Input (IO 303)**

Ruby-MM-416 boards have three I/O ports, each containing eight digital I/O lines. These ports can be configured independently for either input or output. Use a separate driver block for each port. By selecting the digital input driver block for a given port, you configure that port for input.

#### **Scaling Input to Output.**

<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL	double	TTL low = 0.0 TTL high = 1.0

### Driver Block Parameters.

**Channel vector** — Enter numbers between 1 and 8 to select the digital input lines used with this port. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital inputs for the current port, enter

```
[1,2,3,4,5,6,7,8]
```

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — From the list choose A, B, or C.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the base address board setting. For example, if the base address is 300 (hexadecimal), enter

```
0x300
```

### Ruby-MM-416 Digital Output (IO 303)

Ruby-MM-416 boards have three I/O ports, each containing eight digital I/O lines. These ports can be configured independently for either input or output. Use a separate driver block for each port. By selecting the digital input driver block for a given port, that port is configured for input.

#### Scaling Input to Output.

Hardware Input	Block Output Data Type	Scaling
TTL	double	TTL low = 0.0 TTL high = 1.0

**Driver Block Parameters.**

**Channel vector** — Enter numbers between 1 and 8 to select the digital output lines used with this port. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital outputs for the current port, enter

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

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — From the list choose A, B, or C.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the base address board setting. For example, if the base address is 300 (hexadecimal), enter

0x300

## xPC TargetBox IO 304

The Diamond Onyx-MM board has 48 digital I/O lines. They can be programmed for input or output in groups of eight lines. This section includes the following topics:

- “Wiring for IO 304 Test Dongle” on page 5-43
- “Pin Layout IO 304 (Both Connectors)” on page 5-43
- “Testing Model IO 304” on page 5-45
- “Onyx-MM (IO 304) Driver Blocks” on page 5-46

---

**Note** xPC TargetBox does not support the counter/timers and external interrupts on this board.

---

### Wiring for IO 304 Test Dongle

xPC TargetBox communicates with this board using two connectors. Some of the pins in the test dongles are connected together to allow for testing of this board:

- Header (J3) Dongle — The following pins are connected together: 23-39 and 31-47.
- Header (J4) Dongle — The following pins are connected together: 23-39 and 31-47.

### Pin Layout IO 304 (Both Connectors)

The following figure is the pin layout for both of the IO 304 option connectors. These are the J3 and J4 connectors in the manufacturer’s data sheet at <http://www.diamondsystems.com/files/binaries/OMMv14.pdf>.

The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.

Digital I/O Port A	A7	1	2	Gnd
	A6	3	4	Gnd
	A5	5	6	Gnd
	A4	7	8	Gnd
	A3	9	10	Gnd
	A2	11	12	Gnd
	A1	13	14	Gnd
	A0	15	16	Gnd
Digital I/O Port C	C7	17	18	Gnd
	C6	19	20	Gnd
	C5	21	22	Gnd
	C4	23	24	Gnd
	C3	25	26	Gnd
	C2	27	28	Gnd
	C1	29	30	Gnd
	C0	31	32	Gnd
Digital I/O Port B	B7	33	34	Gnd
	B6	35	36	Gnd
	B5	37	38	Gnd
	B4	39	40	Gnd
	B3	41	42	Gnd
	B2	43	44	Gnd
	B1	45	46	Gnd
	B0	47	48	Gnd
	+5V	49	50	Gnd

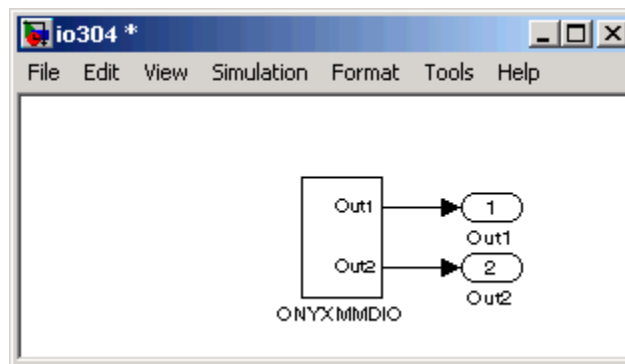
## Testing Model IO 304

A constant value is written to a digital output driver block and then read from a digital input block. The values are compared with an error defined in a second constant block, and the results sent to an Output block:

- 1 In the MATLAB Command Window, type

```
io304
```

The Simulink model for testing the IO 304 option opens.



You can double-click the ONYXMMDIO block to bring up the model.





## Board Characteristics

Board name	Onyx-MM
Manufacturer	Diamond Systems Corporation
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	DIO:Yes
Multiple board support	Yes

## Onyx-MM Digital Input (IO 304)

Onyx-MM boards have two digital I/O chips, each with three 8-bit digital I/O ports, for a total of 48 I/O lines. Each port can be configured independently for either input or output. Use a separate driver block for each port. Select the digital input driver block for a given port to configure the port for input.

### Scaling Input to Output.

Hardware Input	Block Output Data Type	Scaling
TTL	double	TTL low = 0.0 TTL high = 1.0

### Driver Block Parameters.

**Channel vector** — Enter numbers between 1 and 8 to select the digital input lines used with this port. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital inputs for the current port, enter

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

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — From the list choose A, B, or C.

**Chip** — From the list choose 1 or 2.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the base address board setting. For example, if the base address is 300 (hexadecimal), enter

0x300

### Onyx-MM Digital Output (IO 304)

Onyx-MM boards have two digital I/O chips, each with three 8-bit digital I/O ports, for a total of 48 I/O lines. Each port can be configured independently for either input or output. Use a separate driver block for each port. Select the digital output driver block for a given port to configure a port for output.

#### Scaling Input to Output.

Hardware Input	Block Output Data Type	Scaling
TTL	double	TTL low = 0.0 TTL high = 1.0

#### Driver Block Parameters.

**Channel vector** — Enter numbers between 1 and 8 to select the digital output lines used with this port. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital outputs for the current port, enter

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

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — From the list choose A, B, or C.

**Chip** — From the list choose 1 or 2.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the base address board setting. For example, if the base address is 300 (hexadecimal), enter

0x300

## **xPC TargetBox IO 305**

The Diamond Quartz-MM-10 board has 10 counters, eight digital I/O lines, and one interrupt line. Note that this option is no longer available. The MathWorks supplies and supports the drivers and documentation for the customers who already own this option. This section includes the following topics:

- “Wiring for IO 305 Test Dongle” on page 5-50
- “Pin Layout IO 305” on page 5-50
- “Testing Model IO 305” on page 5-52
- “Quartz-MM-10 (IO 305) Driver Blocks” on page 5-53

### **Wiring for IO 305 Test Dongle**

xPC TargetBox communicates with this board using one connector. Some of the pins in the test dongles are connected together to allow for testing of this board.

Header (J3) dongle — The following pins are connected together: 6-17, 12-18, and 43-48.

### **Pin Layout IO 305**

The following figure is the pin layout for the IO 305 option connector. These are the J3 and J4 connectors in the manufacturer’s data sheet at <http://www.diamondsystems.com/files/binaries/QMMv15.pdf>.

The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.

Counter/ Timers	In 1	1	2	In 2	Counter/ Timers	
	Gate 1	3	4	Gate 2		
	Out 1	5	6	Out 2		
	In 3	7	8	In 4		
	Gate 3	9	10	Gate 4		
	Out 3	11	12	Out 4		
	In 5	13	14	Out 5		
	Gate 5	15	16	Fout		
	In 6	17	18	In 7		Counter/ Timers
	Gate 6	19	20	Gate 7		
Out 6	21	22	Out 7			
In 8	23	24	In 9			
Gate 8	25	26	Gate 9			
Out 8	27	28	Out 9			
In 10	29	30	Out 10			
Gate 10	31	32	Interrupt in			
Digital output	Dout 7	33	34	Din 7	Digital input	
	Dout 6	35	36	Din 6		
	Dout 5	37	38	Din 5		
	Dout 4	39	40	Din 4		
	Dout 3	41	42	Din 3		
	Dout 2	43	44	Din 2		
	Dout 1	45	46	Din 1		
	Dout 0	47	48	Din 0		
+5V	49	50	Gnd			

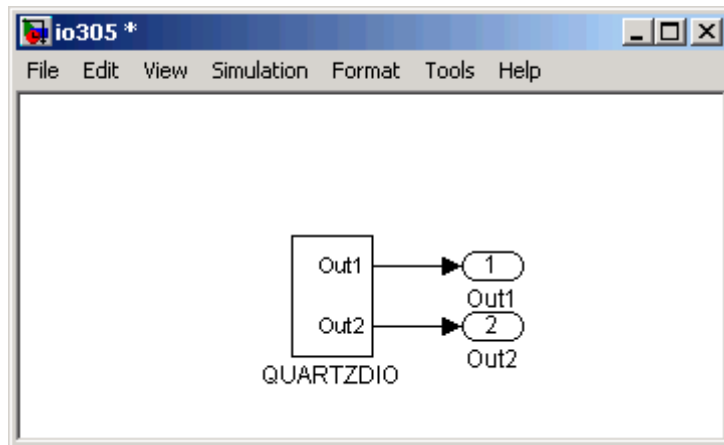
## Testing Model IO 305

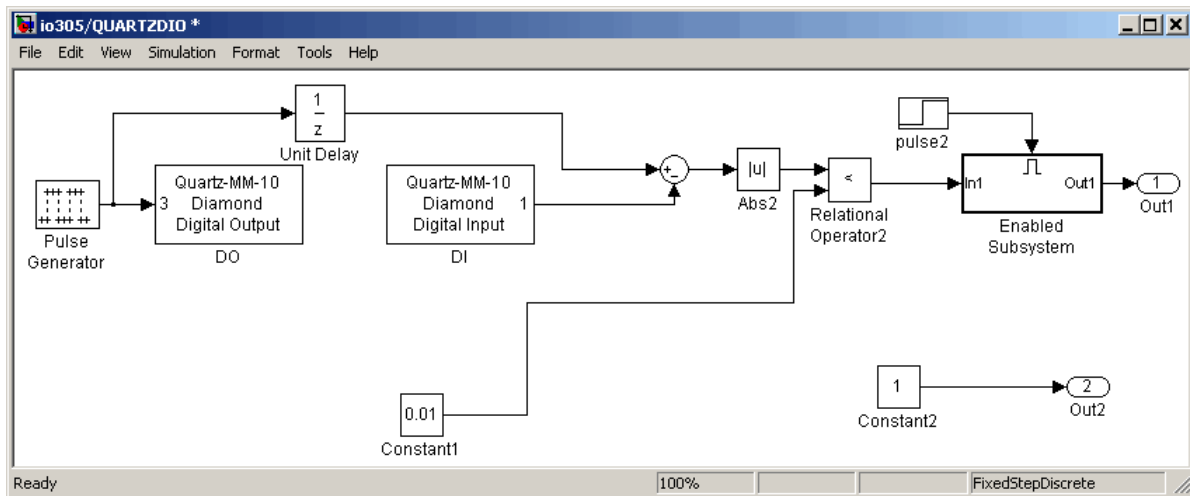
A constant value is written to a digital output driver block and then read from a digital input block. The values are compared with an error defined in a second constant block, and the results sent to an Output block:

- 1 In the MATLAB Command Window, type

```
io305
```

The Simulink model for testing the IO 305 option opens.





## Quartz-MM-10 (IO 305) Driver Blocks

The Quartz-MM-10 has eight digital input lines, eight digital output lines, and 10 counter/timers.

xPC TargetBox supports this board with these driver blocks:

- “Quartz-MM-10 Digital Input (IO 305)” on page 5-54
- “Quartz-MM-10 Digital Output (IO 305)” on page 5-55
- “Quartz-MM-10 Counter PWM (IO 305)” on page 5-56
- “Quartz-MM-10 Counter PWM & ARM (IO 305)” on page 5-57
- “Quartz-MM-10 Counter FM (IO 305)” on page 5-59
- “Quartz-MM-10 Counter FM & ARM (IO 305)” on page 5-60
- “Quartz-MM-10 PWM Capture (IO 305)” on page 5-62
- “Quartz-MM-10 FM Capture (IO 305)” on page 5-63
- “Quartz-MM (IO 305)” on page 5-63

**Board Characteristics**

Board name	Quartz-MM-10
Manufacturer	Diamond Systems Corporation
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

**Quartz-MM-10 Digital Input (IO 305)**

**Scaling Input to Output.**

Hardware Input	Block Output Data Type	Scaling
TTL	double	TTL low = 0.0 TTL high = 1.0

**Driver Block Parameters.**

**Channel vector** — Enter numbers between 1 and 8 to select the digital input lines used with this port. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital inputs for one port, enter

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

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Sample time** — Enter a base sample time or a multiple of the base sample time.



**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

## Quartz-MM-10 Digital Output (IO 305)

### Scaling Input to Output.

Hardware Output	Block Input Data Type	Scaling
TTL	double	< 0.5 = TTL low > 0.5 = TTL high

### Driver Block Parameters.

**Channel vector** — Enter numbers between 1 and 8 to select the digital input lines used with this port. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital inputs for one port, enter

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

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

### Quartz-MM-10 Counter PWM (IO 305)

The Quartz-MM-10 has two AM9513A chips with five counters each.

The Quartz-MM-10 PWM driver programs the AM9513A for PWM (pulse width modulation) signal generation (a square wave with fixed frequency and variable duty cycle). The block has one input that defines the variable duty cycle between 0 and 1. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

#### Scaling Input to Output.

Hardware Output	Block Input Data Type	Scaling
TTL	double	0 to 1

#### Driver Block Parameters.

**Counter** — From the list, choose 1, 2, 3, 4, or 5 to select which counter is used with this driver block. In each case, one block is needed for each counter.

**Frequency base** — From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

**Relative output frequency** — Enter a value between 0 and  $.25 * \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM signal.

For example, if the output frequency of a square wave must be 17.5 kHz, then choose F2=400kHz as the **Frequency base** and enter 0.04375 as the **Relative output frequency**.  $400 \text{ kHz} \times 0.04375 = 17.5 \text{ kHz}$

**Level sequence of square wave** — From the list, choose either high-low or low-high.

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

**Level when disarmed** — From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and becomes armed when the application begins running. This parameter sets the TTL level when the counter is disarmed.

**Sample time** — Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle).

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

### Quartz-MM-10 Counter PWM & ARM (IO 305)

The Quartz-MM-10 has two AM9513A chips with five counters.

The Quartz-MM-10 PWM & ARM driver programs the AM9513A for PWM or disarmed signal generation (a square wave with fixed frequency and variable duty cycle). Additionally the driver allows you to arm and disarm the counter by using the second block input. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

#### Scaling Input to Output.

Hardware Output	Block Input Data Type	Scaling
TTL	Duty cycle: double Arm: double	0 to 1 <0.5 disarmed >0.5 armed

#### Driver Block Parameters.

**Counter** — From the list, choose 1, 2, 3, 4, or 5 to select which counter is used with this driver block. In each case, one block is needed for each counter.

**Frequency base** — From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

**Relative output frequency** — Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM signal.

For example, if the output frequency of a square wave must be 17.5 kHz, then choose F2=400kHz as the **Frequency base** and enter 0.04375 as the **Relative output frequency**.  $400 \text{ kHz} \times 0.04375 = 17.5 \text{ kHz}$

**Level sequence of square wave** — From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

**Level when disarmed** — From the list, choose either high or low. The counter is automatically disarmed when the target application is not running. If the application is running, the second input port controls whether the counter is armed or disarmed. This parameter sets the TTL level when the counter is disarmed.

**Sample time** — Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle).

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

## Quartz-MM-10 Counter FM (IO 305)

The Quartz-MM-10 has two AM9513A chip with five counters each.

The Quartz-MM-10 FM driver programs the AM9513A for FM (frequency modulation) signal generation (a square wave with fixed duty cycle and variable frequency). For the corresponding counter channel, the FM signal is output at the pin named OUT.

### Scaling Input to Output.

Hardware Output	Block Input Data Type	Scaling
TTL	double	0 to 1

### Driver Block Parameters.

**Counter** — From the list, choose 1, 2, 3, 4, or 5 to select which counter is used with this driver block. In each case, one block is needed for each counter.

**Frequency base** — From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

**Output duty cycle** — Enter a value between 0 and 1 to set the duty cycle of the square wave. The duty cycle is held fixed during execution of the target application.

---

**Note** Your control over the output duty cycle is not always precise. See PWM and FM Driver Block Notes in xPC Target I/O Reference for details.

---

**Level sequence of square wave** — From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

**Level when disarmed** — From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and becomes armed when the application begins running. This parameter sets the TTL level when the counter is disarmed.

**Sample time** — Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle).

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

### **Quartz-MM-10 Counter FM & ARM (IO 305)**

The Quartz-MM-10 has two AM9513A chips with five counters each.

The Quartz-MM-10 FM & ARM driver programs the AM9513A for FM (frequency modulation) signal generation (a square wave with fixed duty cycle and variable frequency). Additionally the driver allows you to arm and disarm the counter by the second block input. For the corresponding counter channel, the FM signal is output at the pin named OUT.

#### **Scaling Input to Output.**

<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
TTL	Variable frequency: double Arm: double	<0.5 disarmed ≥0.5 armed

#### **Driver Block Parameters.**

**Counter** — From the list, choose 1, 2, 3, 4, 5, 6, 7, 8, 9, or 10 to select which counter is used with this driver block. In each case, one block is needed for each counter.

**Frequency base** — From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

**Output duty cycle** — Enter a value between 0 and 1 to set the duty cycle of the square wave. The duty cycle is held fixed during execution of the target application.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” in xPC Target I/O Reference for details.

---

**Level sequence of square wave** — From the list, choose either high-low or low-high.

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the **Output duty cycle** defined in the setting above defines the duration of the TTL high part.

**Level when disarmed** — From the list, choose either high or low. The counter is automatically disarmed when the target application is not running. If the application is running, the second input port controls whether the counter is armed or disarmed. This parameter sets the TTL level when the counter is disarmed.

**Sample time** — Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle).

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

**Quartz-MM-10 PWM Capture (IO 305)**

This block programs the AM9513A for capturing PWM signals by using two counters. One counter measures the cycle duration, and the other counter measures the length of time the signal is high.

There are two outputs. One output is the relative frequency compared to the base frequency. The other output is the duty cycle. To get the actual frequency, multiply the base frequency by the relative frequency.

The PWM signal must enter the pins of both corresponding counter channels (parallel wiring) named GATE. Both CLK pins must be left unconnected.

**Scaling Input to Output.**

Hardware Output	Block Input Data Type	Scaling
TTL	double	0 to 1

**Driver Block Parameters.**

**Counter** — From the list, choose 1&2, 2&3, 3&4, 4&5, 6&7, 7&8, 8&9, or 9&10. This selects which two counters the driver block uses to determine the PWM. Use one block for each pair of counters. No two blocks should use the same counter. For example, use counters 1&2 for one block and 3&4 for a second block. Do not use a combination like 1&2 and 2&3.

**Frequency base** — From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

**Sample time** — Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle).

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



## Quartz-MM-10 FM Capture (IO 305)

This block programs the AM9513A for capturing FM signals.

There is one output for relative frequency compared to the base frequency. To get the actual frequency, multiply the base frequency by the relative frequency.

The FM signal must enter the pin of the corresponding counter channel named GATE. Leave the CLK pin unconnected.

### Scaling Input to Output.

Hardware Output	Block Input Data Type	Scaling
TTL	double	0 to 1

### Driver Block Parameters.

**Counter** — From the list, choose 1, 2, 3, 4, 5, 6, 7, 8, 9, or 10. This selects which counter the driver block uses to determine the FM. In each case, one block is needed for each counter.

**Frequency base** — From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

**Sample time** — Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle).

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

## Quartz-MM (IO 305)

You can use this block to program the AM9513A counter. The PWM, PWM & ARM, FM, FM & ARM, PWM Capture, and FM Capture blocks use this block in their underlying subsystems. The API for this block is not currently documented.

## xPC TargetBox IO 306

The Real Time Devices DM6814 board has three encoders, 12 digital I/O lines, and two interrupts. This section includes the following topics:

- “Wiring for IO 306 Test Dongle” on page 5-64
- “Pin Layout IO 306 (1 of 2 Connectors)” on page 5-64
- “Pin Layout IO 306 (2 of 2 Connectors)” on page 5-66
- “Pin Layout and Ports” on page 5-68
- “Testing Model IO 306” on page 5-68
- “DM6814 (IO 306) Driver Blocks” on page 5-70

---

**Note** xPC TargetBox does not support the counter/timers on this board related to the 8254 counter chip. It does support the counters required for incremental encoding.

---

### Wiring for IO 306 Test Dongle

xPC TargetBox communicates with this board using two connectors. Some of the pins in the test dongle are connected together to allow for testing of this board.

Header (P2) dongle — The following pins are connected together: 5-13, 7-15, 21-29, 23-31, 37-45, and 39-47.

---

**Note** Header (P3) — While this header is connected to an I/O connector and shipped with a terminal board and cable, the xPC TargetBox is not shipped with a dongle attached to this I/O connector for testing.

---

### Pin Layout IO 306 (1 of 2 Connectors)

The following figure is the pin layout for the IO 306 option first connector. This is the P2 connector on the manufacturer’s data sheet [http://www.rtdusa.com/NEW\\_manuals/hardware/datamodules/dm6814.pdf](http://www.rtdusa.com/NEW_manuals/hardware/datamodules/dm6814.pdf).

The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.

Encoder	Inc Enc 3	IRQin	1	2	Ext int1		
		Overflow 3	Out	3	4	Dgnd	
		Inc Enc 3	ChB	5	6	Dgnd	
		Inc Enc 3	ChA	7	8	Dgnd	
	D In	P4.3		9	10	Dgnd	
				11	12	Dgnd	
	D In/Out	P4.1		13	14	Dgnd	
				15	16	Dgnd	
	Encoder	Inc Enc 2	IRQin	17	18	Dgnd	
			Overflow 3	Out	19	20	Dgnd
			Inc Enc 2	ChB	21	22	Dgnd
			Inc Enc 2	ChA	23	24	Dgnd
		D In	P2.3		25	26	Dgnd
					27	28	Dgnd
		D In/Out	P2.1		29	30	Dgnd
					31	32	Dgnd
Encoder		Inc Enc 1	IRQin	33	34	Dgnd	
			Overflow 3	Out	35	36	Dgnd
			Inc Enc 1	ChB	37	38	Dgnd
			Inc Enc 1	ChA	39	40	Dgnd
		D In	P0.3		41	42	Dgnd
					43	44	Dgnd
		D In/Out	P0.1		45	46	Dgnd
					47	48	Dgnd
			+5V	49	50	Dgnd	

## **Pin Layout IO 306 (2 of 2 Connectors)**

The following figure is the pin layout for the IO 306 option second connector. This is the P3 connector on the manufacturer's data sheet <http://www.rtdusa.com/manuals/DM/dm5810.pdf>.

The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.

	NC	1	2	Ext int2
	NC	3	4	Dgnd
	NC	5	6	Dgnd
	NC	7	8	Dgnd
D In	P5.3	9	10	Dgnd
	P5.2	11	12	Dgnd
	P5.1	13	14	Dgnd
	P5.0	15	16	Dgnd
	NC	17	18	Dgnd
	NC	19	20	Dgnd
	NC	21	22	Dgnd
	NC	23	24	Dgnd
D In	P3.3	25	26	Dgnd
	P3.2	27	28	Dgnd
	P3.1	29	30	Dgnd
	P3.0	31	32	Dgnd
	NC	33	34	Dgnd
	NC	35	36	Dgnd
	NC	37	38	Dgnd
	NC	39	40	Dgnd
D In	P1.3	41	42	Dgnd
	P1.2	43	44	Dgnd
	P1.1	45	46	Dgnd
	P1.0	47	48	Dgnd
	+5V	49	50	Dgnd

## Pin Layout and Ports

This section describes the association of the three digital I/O ports (A, B, and C) with the P2 and P3 connector pins.

The DM6814 board has 24 digital I/O lines organized into three 8-bit ports, A, B, and C. Each port has four lines brought out on connector P2 and four on connector P3.

Port	Channels	Correspond to Pins	On Connector
A	1-4	P0.0, P0.1, P0.2, P0.3	P2
	5-8	P1.0, P1.1, P1.2, P1.3	P3
B	1-4	P2.0, P2.1, P2.2, P2.3	P2
	5-8	P3.0, P3.1, P3.2, P3.3	P3
C	1-4	P4.0, P4.1, P4.2, P4.3	P2
	5-8	P5.0, P5.1, P5.2, P5.3	P3

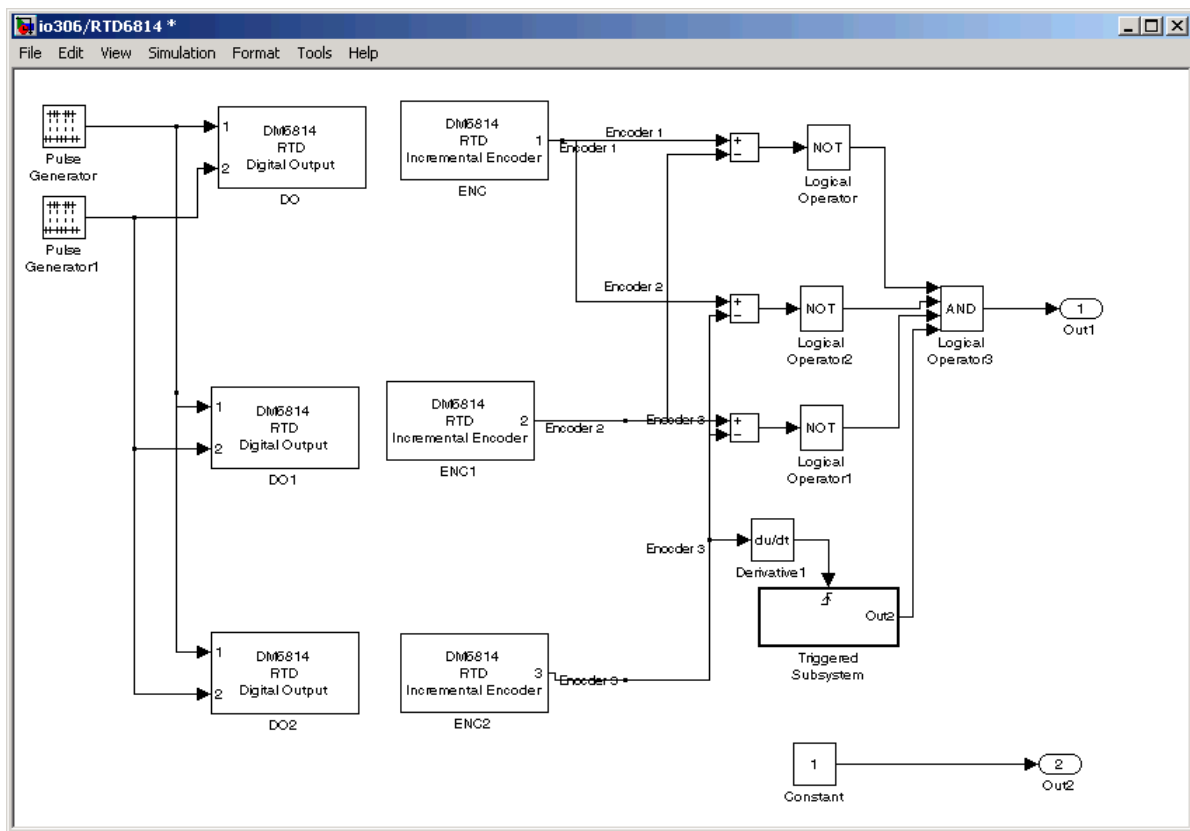
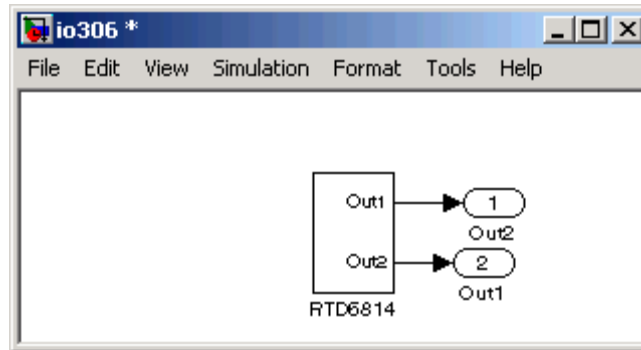
You can use any subset of the 24 channels for input. You can only use channels 1 and 2 on each port for output (a maximum of six output channels). You cannot use any channel simultaneously for input and output.

## Testing Model IO 306

**1** In the MATLAB Command Window, type

```
io306
```

The Simulink model for testing the IO 306 option opens.



## DM6814 (IO 306) Driver Blocks

The DM6814 is a 16-bit counting board with three channels. This board typically connects to incremental encoders. Incremental encoders convert physical motion into electrical pulses that can be used to determine velocity, direction, and distance.

Each board has three I/O ports, with each port containing eight digital I/O lines.

xPC TargetBox supports this board with these driver blocks:

- “DM6814 Incremental Encoder (IO 306)” on page 5-70
- “DM6814 Digital Input (IO 306)” on page 5-71
- “DM6814 Digital Output (IO 306)” on page 5-72

---

**Note** xPC TargetBox does not support the 12 digital input lines on this board.

---

### Board Characteristics

Board name	DM6814
Manufacturer	Real Time Devices
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

### DM6814 Incremental Encoder (IO 306)

#### Driver Block Parameters.

**Encoder channel** — From the list choose, 1, 2, or 3. This parameter specifies which channel you use for this block. For the same board (same base address) two blocks cannot have the same channel number.



**Counter initial value** — Enter the initial value of the counter. The value must be between 1 and  $2^{16} - 1$ .

**Enable counter reset on px.2 (index input)** — If this check box is selected, the counter is reset to its initial value (default zero) whenever the incremental encoder is moved over its index mark.

The px.2 pins correspond to the even ports. For this purpose you must connect the incremental encoder index output to the correct pin: for channel 1, pin 43, P0.2; for channel 2, pin 27, P2.2; for channel 3, pin 11, P4.2.

**Sample time** — Base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

### DM6814 Digital Input (IO 306)

Of the board's three I/O ports, you can use only one digital input block per port. You can use an input and an output block for the same port, but each must use a different channel.

#### Scaling Input to Output.

Hardware Input	Block Output Data Type	Scaling
TTL	double	TTL low = 0.0 TTL high = 1.0

#### Driver Block Parameters.

**Channel Vector** — Enter numbers between 1 and 8 to select the digital input lines used with this port. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines used.

For example, to use all the digital inputs for the current port, enter

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

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — Select port A, B, or C.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the base address board setting. For example, if the base address is 300 (hexadecimal), enter

0x300

### **DM6814 Digital Output (IO 306)**

Of a port's eight digital I/O lines, you can configure two for output. You can use only one digital input block per port. You can use an input and an output block for the same port, but each must use a different channel.

#### **Scaling Input to Output.**

<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
TTL	double	<0.5 = TTL low ≥0.5 = TTL high

#### **Driver Block Parameters.**

**Channel Vector** — Enter either 1 or 2 to select the digital output lines you use with this port. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines used.

For example, to use all available digital outputs for the current port, enter

[1,2]

Number the lines beginning with 1 even if the board manufacturer starts numbering the lines with 0.

**Port** — Select port A, B, or C.

**Reset vector** — The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is used for all channels. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 0, the channel remains at the last value attained while the model was running.

**Initial value vector** — The initial value vector contains the initial values (0 or 1) of the output channels. Enter a scalar or a vector that is the same length as the channel vector. If you enter a scalar, that value is used for all channels. The channels are set to these initial values between the time the model is downloaded and the time it is started.

**Sample time** — Enter a base sample time or a multiple of the base sample time.

**Base address** — Enter the base address of the board. This entry must correspond to the base address board setting. For example, if the base address is 300 (hexadecimal), enter

0x300

## xPC TargetBox IO 308

The Softing CAN-AC2-104 board has two CAN channels. This section includes the following topics:

- “Wiring for IO 308 Test Dongle” on page 5-74
- “Pin Layout IO 308” on page 5-74
- “Testing Model IO 308” on page 5-75
- “CAN-AC2-104 Driver Blocks (IO 308)” on page 5-76
- “CAN-AC2-104 FIFO Driver Blocks (IO 308)” on page 5-86

### Wiring for IO 308 Test Dongle

xPC TargetBox communicates with this board using two CAN serial connectors.

The test dongle is a CAN serial cable, with termination resistors, connected between the two CAN 9-pin connectors.

---

**Note** The IO 308 option is shipped with terminal resistors enabled. When you connect this option to a CAN field bus, you need to add terminal resistors to your cable.

---

### Pin Layout IO 308

The following figure is the pin layout for both of the IO 308 option connectors. The pin layout for the D-sub 9 connector is the same for both CAN channels.

The numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below. However, the pin numbers on the terminal board are numbered sequentially.

Drain	5	9	NC
NC	4	8	NC
Gnd	3	7	CAN_H
CAN_L	2	6	Gnd
NC	1		

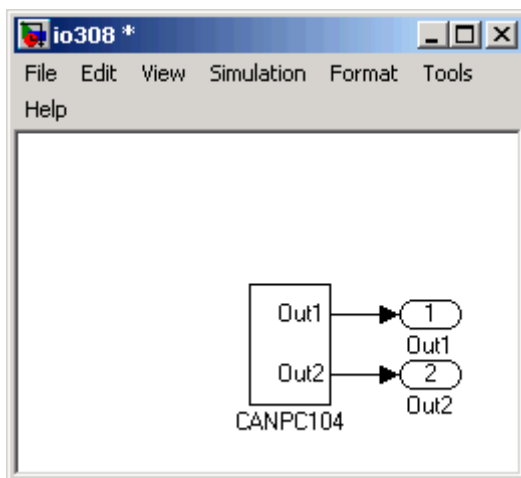
## Testing Model IO 308

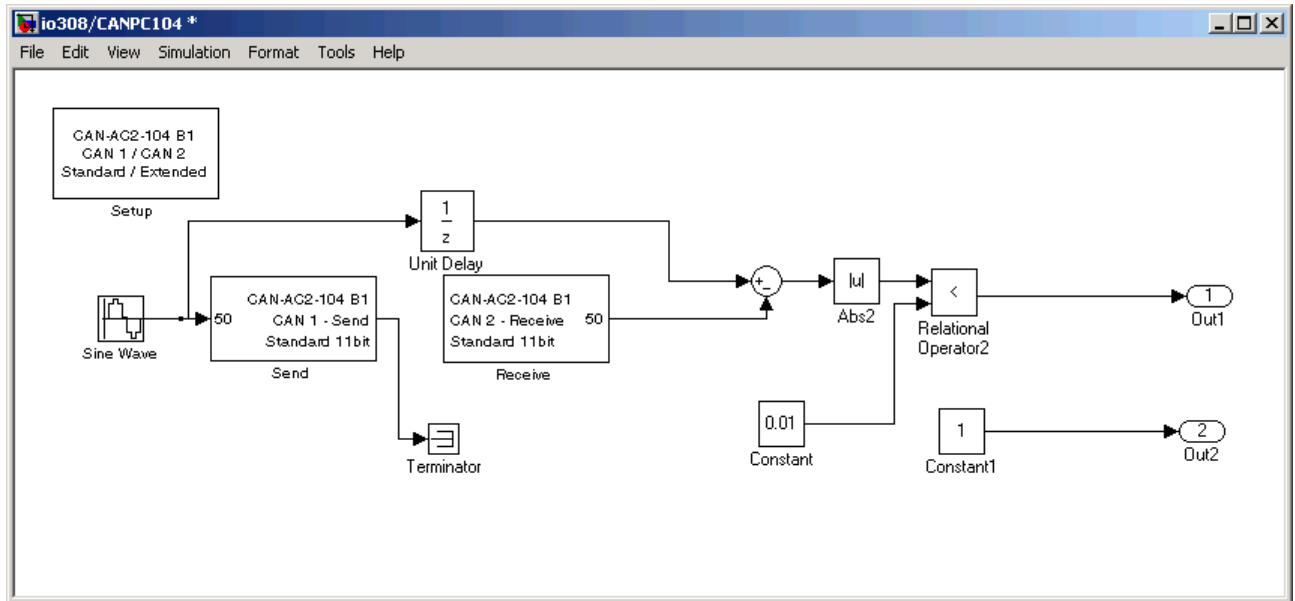
A sine wave signal is written to a CAN output driver block and then read from a CAN input block. After delaying the input signal, the values are compared with an error defined in a second Constant block, and the results sent to an Output block:

- In the MATLAB Command Window, type

```
io308
```

The Simulink model for testing the IO 308 option opens.





## CAN-AC2-104 Driver Blocks (IO 308)

The driver blocks described here support the CAN-AC2-104 (PC/104). This board uses the Philips SJA1000 chip as the CAN controller in this configuration. It supports both standard and extended identifier ranges in parallel.

For more information about using CAN with xPC Target, see the chapter, “CAN I/O Support for FIFO” in xPC Target I/O Reference.

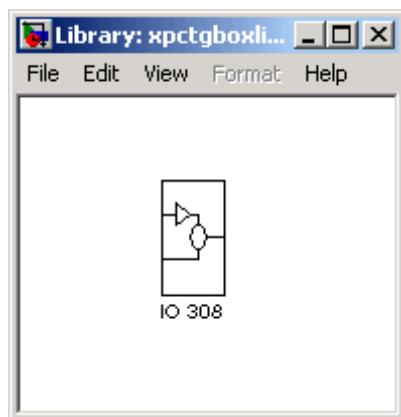
1 In the MATLAB Command Window, type

```
xpctgboxlib
```

The xPC TargetBox library dialog box opens.

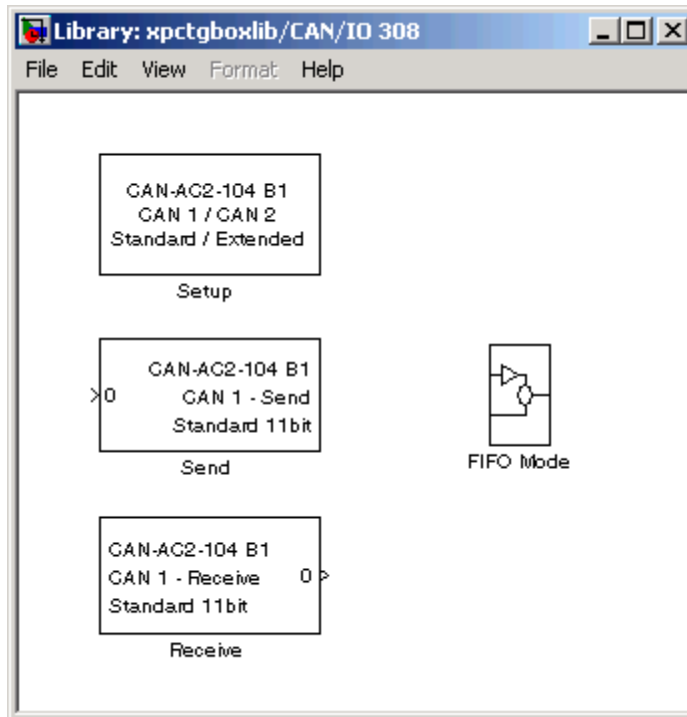
- 2 Double-click the block labeled CAN.

The CAN library for the xPC TargetBox opens. Currently there is only one supported CAN board for xPC TargetBox.



3 Click the block labeled **IO 308**.

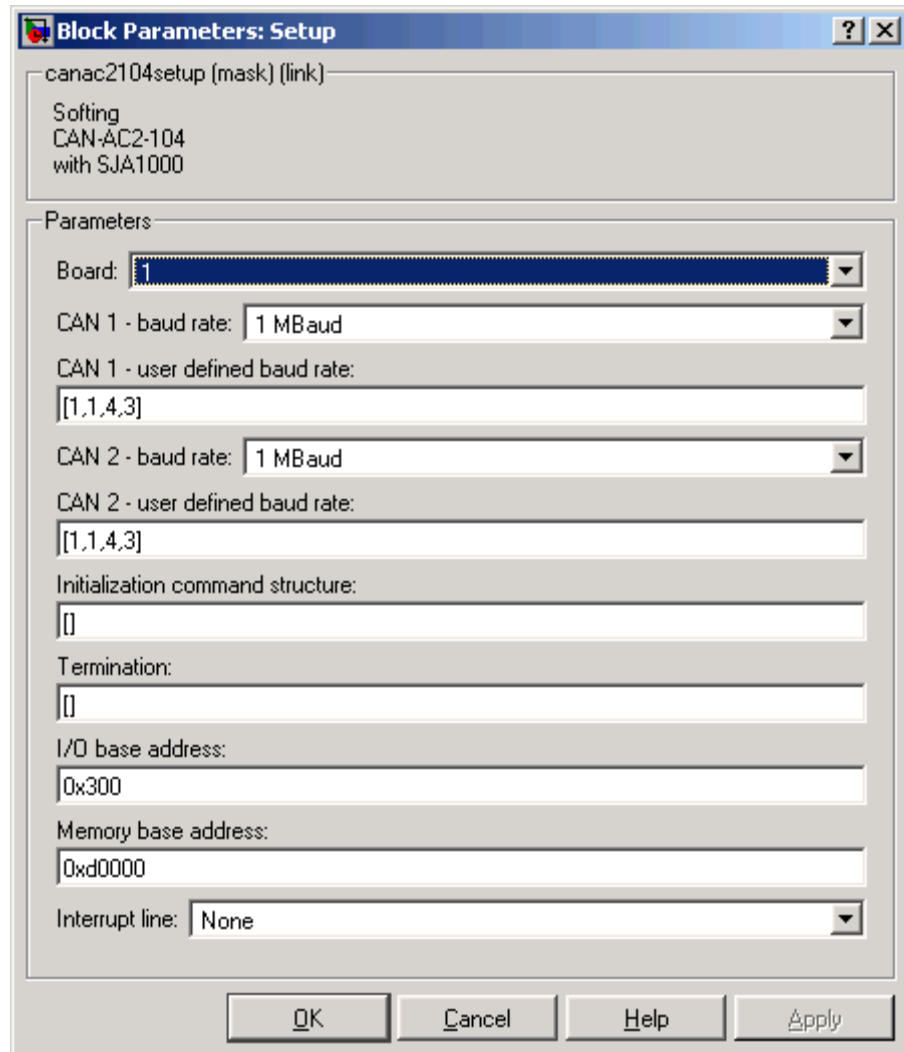
The CAN IO 308 dialog box opens with the three available CAN blocks: Setup, Send, and Receive. The fourth block group contains the FIFO mode subgroup.





## Setup Driver Block (IO 308)

The Setup block is used to define general settings of the stacked CAN boards. The CAN driver blocks for this board support up to three boards for each target system, which leads to the availability of up to six CAN ports. For each board in the target system, you must use exactly one Setup driver block in a model.



The dialog box of the Setup block lets you define the following settings:

**Board** — Defines which board is being accessed by this driver block instance. The board number (1...3) can be seen as a reference identifier to differentiate the boards if multiple boards are present in the target system. The physical board finally referenced by the board number depends on the edit field described below. If just one board is present in the target system, select board number 1.

**CAN 1 - baud rate** — Defines the most common baud rates for CAN port 1. If special timing is necessary (baud rate), the value **CAN 1 - user defined baud rate** can be selected. In this case the third control (edit field) is used to provide the four values for the timing information. The vector elements have the following meaning:

```
[ Prescalar, Synchronization-Jump-Width, Time-Segment-1,  
  Time-Segment-2 ]
```

For more information about these values, see the Softing user manual for this board.

**CAN 2 - baud rate** — Defines the most common baud rates for CAN port 2. If special timing is necessary (baud rate), you can select the value **User defined**. In this case the fifth control (edit field) is used to provide the four values for the timing information. The vector elements have the following meanings:

```
[ Prescalar, Synchronization-Jump-Width, Time-Segment-1,  
  Time-Segment-2 ]
```

For more information about these values, see the Softing user manual for this board.

**Initialization and Termination** — Defines CAN messages sent during initialization and termination of the Setup block.

**I/O base address** — Defines the I/O base address of the board to be accessed by this block instance. The I/O base address is given by the DIP switch setting on the board itself. The I/O address range is 3 bytes and is mainly used to transfer which memory base address the board should use. See the Softing user manual for this board on how you can set the I/O base address. The I/O base address entered in this control must correspond with the DIP switch

setting on the board. If more than one board is present in the target system, a different I/O base address must be entered for each board. In this case the I/O base address itself defines which board is referenced by which board number.

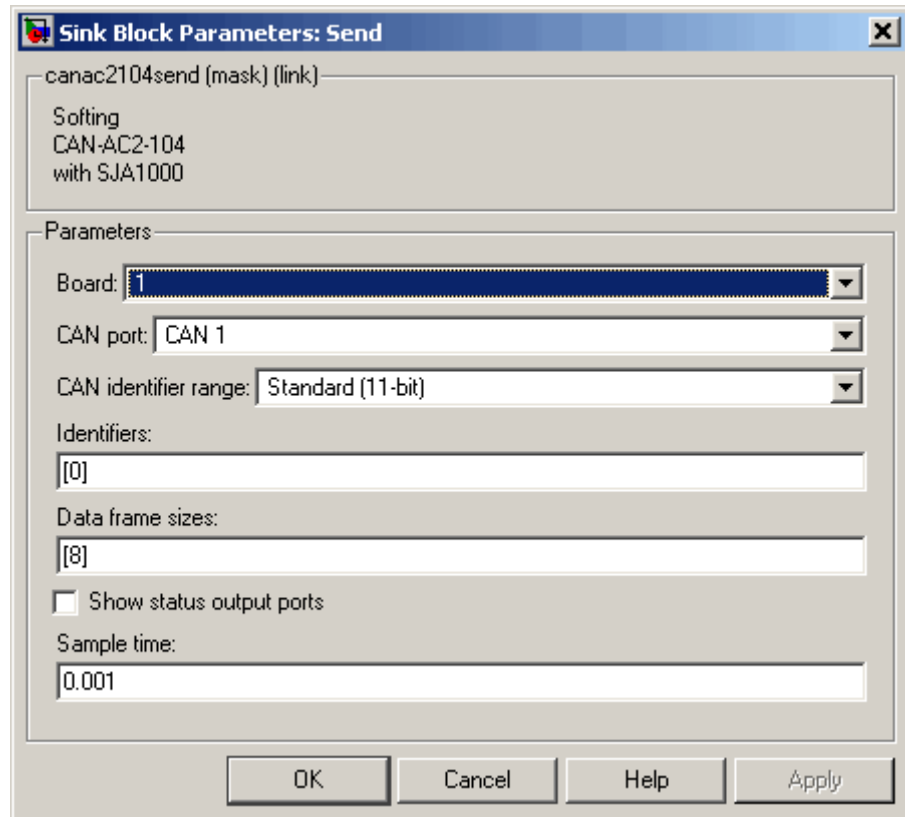
**Memory base address** — Defines the memory base address of the board to be accessed by this block instance. The memory base address is a software setting only (no corresponding DIP switch is found on the board). The memory address range is 4 kilobytes. If more than one board is present in the target system, a different memory base address must be entered for each board. You have to make sure that the defined address ranges do not overlap. Because the xPC Target kernel only reserves a subset of the address range between 640 kilobytes and 1 MB for memory mapped devices, the address ranges must be within the following range:

C0000-DC000

The board allows activating proper termination for each of the two CAN ports separately by means of jumpers found on the board. Refer to the board user manual for how the DIP switches have to be set. Both CAN ports have to be terminated properly when you use the provided model to test the board and drivers.

## Send Driver Block (IO 308)

The Send driver block is used to transmit data to a CAN network from within a block model.



The dialog box of the Send block lets you define the following settings:

**Board** — Defines which board to use to send out the CAN messages defined by this block instance. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

**CAN port** — Selects the CAN port to send the CAN message out.

**CAN identifier range** — Selects the identifier range of the CAN-messages sent out by this block instance. If an application makes use of mixed standard and extended identifier ranges, at least two instances of this block have to be used, each defining the corresponding identifier range.

**Identifiers** — Defines the identifiers of the CAN messages sent out by this block. It must be a row vector where the elements define a set of either standard or extended identifiers. Each element must be in the range between 0 and 2031 for standard identifiers or 0 and  $2^{29} - 1$  for extended identifiers. The number of identifiers for each CAN port in a model per physical CAN board cannot exceed 200. The number of elements defined here also defines the number of input ports of the block. The block icon displays the selected identifier at each input port. Each input port accepts the data frame to be sent along with the CAN message. The signal entering each input port must be a scalar of type `double` representing the maximum size of 8 bytes of a CAN message data frame.

**Data frame sizes** — Defines the data frame size for each identifier (CAN message) in bytes. It must be a row vector where the elements define a set of data frame sizes. Each element must be in the range between 1 and 8. If the data frame sizes for all identifiers defined in the preceding control have to be the same, the size can be provided as a scalar only and scalar expansion applies. If the sizes are different for at least two identifiers (CAN messages), one size element must be provided for each identifier specified in the **Identifier** control. Therefore the lengths of the two vectors must be the same.

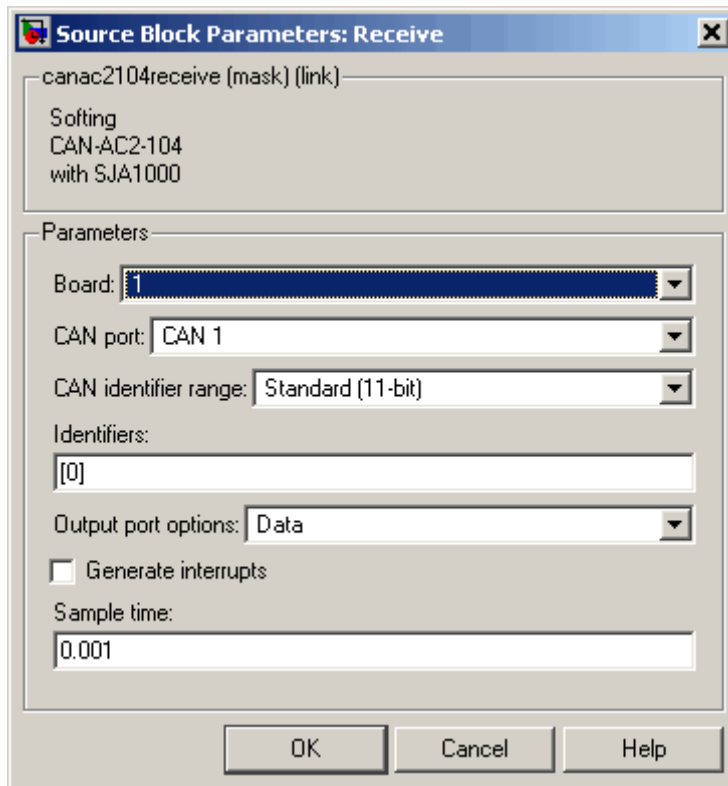
**Show status output ports** — Lets you enable status output ports for each identifier (CAN message). If the check box is selected, the block shows as many output ports as input ports. The data type of each output port is a `double` and the value is identical to the return argument of function `CANPC_write_object(...)`, described in the Softing user manual. Refer to the manual for more information.

**Sample time** — Defines the sample time at which the Send block is executed during a model (target application) run.

You can use as many instances of the Send block in the model as needed. For example by using two instances of the block, you can define different sample times at which CAN messages are sent out. Or you can use multiple instances to structure your model more efficiently.

## Receive Driver Block (IO 308)

The Receive driver block is used to retrieve data from a CAN network to be used within a block model. You can use as many instances of the Receive block in the model as needed. For example, by using two instances of the block with different sample times, you can retrieve CAN messages at different rates or you can use multiple instances to structure your model more efficiently.



The dialog box of the block lets you define the following settings.

**Board** — Defines the board from which the CAN messages defined by this block instance are to be retrieved. For more information about the meaning of the board number, see the Setup driver block. If just one board is present in the target system, select board number 1.

**CAN port** — Selects the CAN port from which to send the CAN message out.

**CAN identifier range** — Selects the identifier range of the CAN messages retrieved by this block instance. If an application makes use of mixed standard and extended identifier ranges, at least two instances of this block have to be used, each defining the corresponding identifier range.

**Identifiers** — Specifies the identifiers of the CAN messages retrieved by this block. It must be a row vector where the elements define a set of either standard or extended identifiers. Each element must be in the range between 0 and 2031 for standard identifiers, or 0 and  $2^{29} - 1$  for extended identifiers. The number of identifiers for each CAN port in a model per physical CAN board cannot exceed 200. The number of elements defined here defines the number of output ports of the block. The block icon displays the selected identifier at each output port. Each output port outputs the data frame being retrieved along with the CAN message. The signal leaving each output port is a scalar of type double representing the maximum size of 8 bytes of a CAN message data frame.

**Output port options** — Defines which type of retrieved data is output at each output port. Three different types of data can be output: data frame, status, and timestamp. The status information is of type double and is identical to the return value of function `CANPC_read_rcv_data(...)`, described in the Softing user manual. Refer to the manual for more information. The timestamp information is of type double and outputs the most recent time at which a CAN message with the corresponding identifier has been received. This time information in seconds (with a resolution of 1 microsecond) can be used to implement timeout logic within your model.

The pop-up menu lets you select which output information is output at each output port of the block. If Data is selected, each output port signal is a scalar only. If Data-Status is selected, each output port signal is a vector with two elements, where the first element contains the data frame and the second element the status information. If Data-Status-Timestamp is selected, each output port signal is a vector with three elements, where the first element contains the data frame, the second element the status information, and the third element the timestamp.

**Generate interrupts** — Defines whether the CAN messages defined in this instance of the block initiate an interrupt from the CAN board each time they are received. If selected, this check box allows controlling the model (target application) execution with CAN messages.

**Sample time** — Defines the sample time at which the Receive block is executed during a model (target application) run.

### **CAN-AC2-104 FIFO Driver Blocks (IO 308)**

The driver blocks described here support the CAN-AC2-104 (PC/104) using FIFO mode. The Philips SJA1000 chip is used as the CAN controller in this configuration. It supports both standard and extended identifier ranges in parallel. The driver blockset for this board is found in the xPC TargetBox I/O block library in the group CAN/Softing.

For more information about using CAN with xPC TargetBox, see “CAN I/O Support for FIFO” in xPC Target I/O Reference.

**1** In the MATLAB Command Window, type

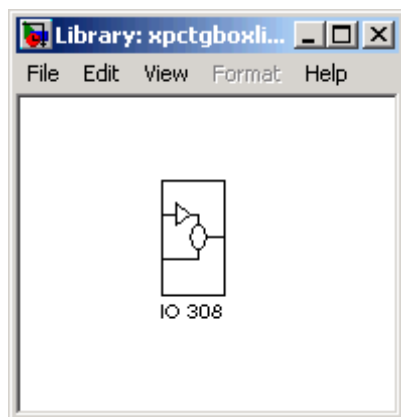
```
xpctgboxlib
```

The xPC TargetBox library dialog box opens.



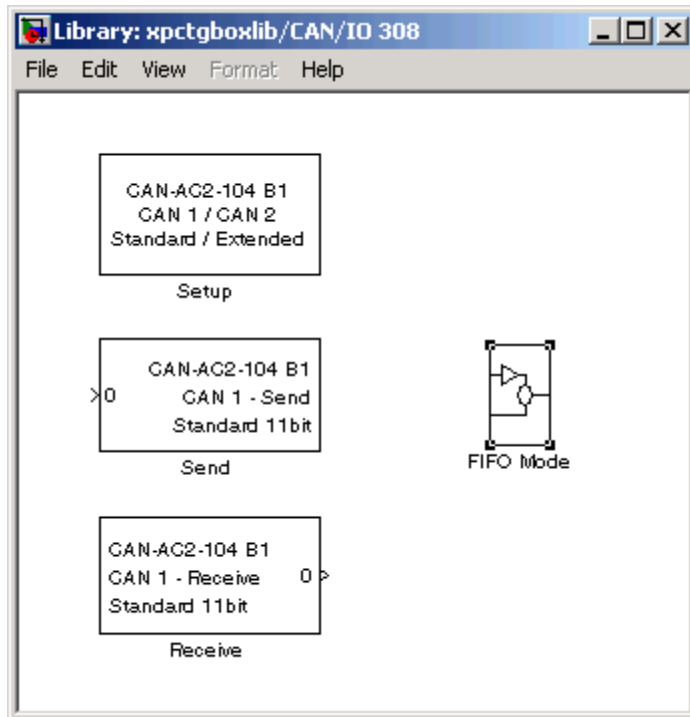
**2** Click the block labeled CAN.

The CAN library for the xPC TargetBox opens. Currently there is only one supported CAN board for xPC TargetBox.



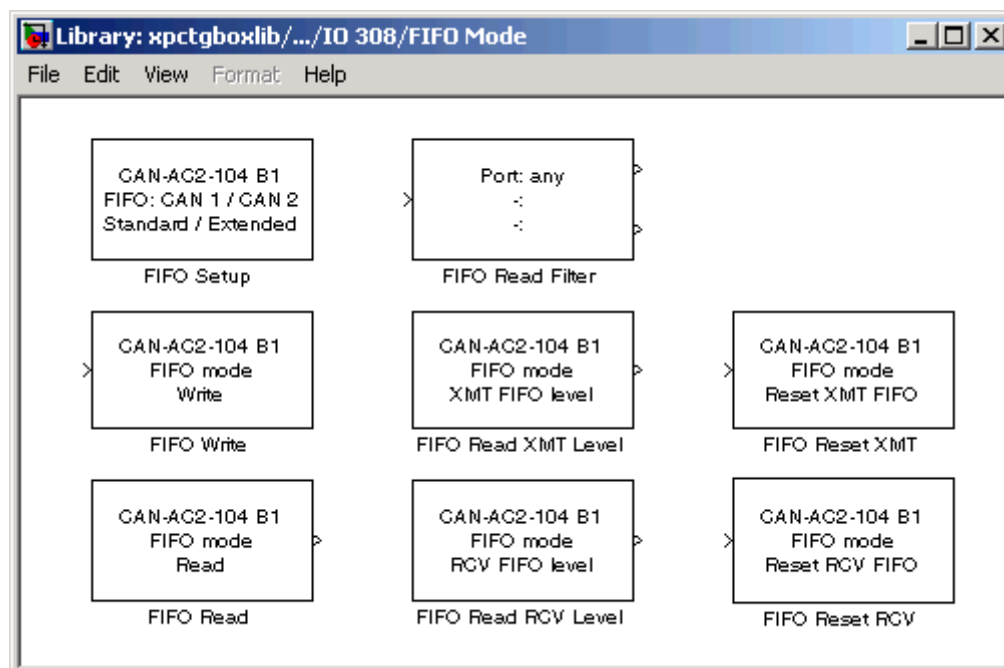
- 3 Click the block labeled IO 308.

The CAN / IO 308 dialog box opens with the three available CAN blocks: Setup, Send, and Receive. The fourth block group contains the FIFO Mode subgroup.



**4** Click the block labeled FIFO Mode.

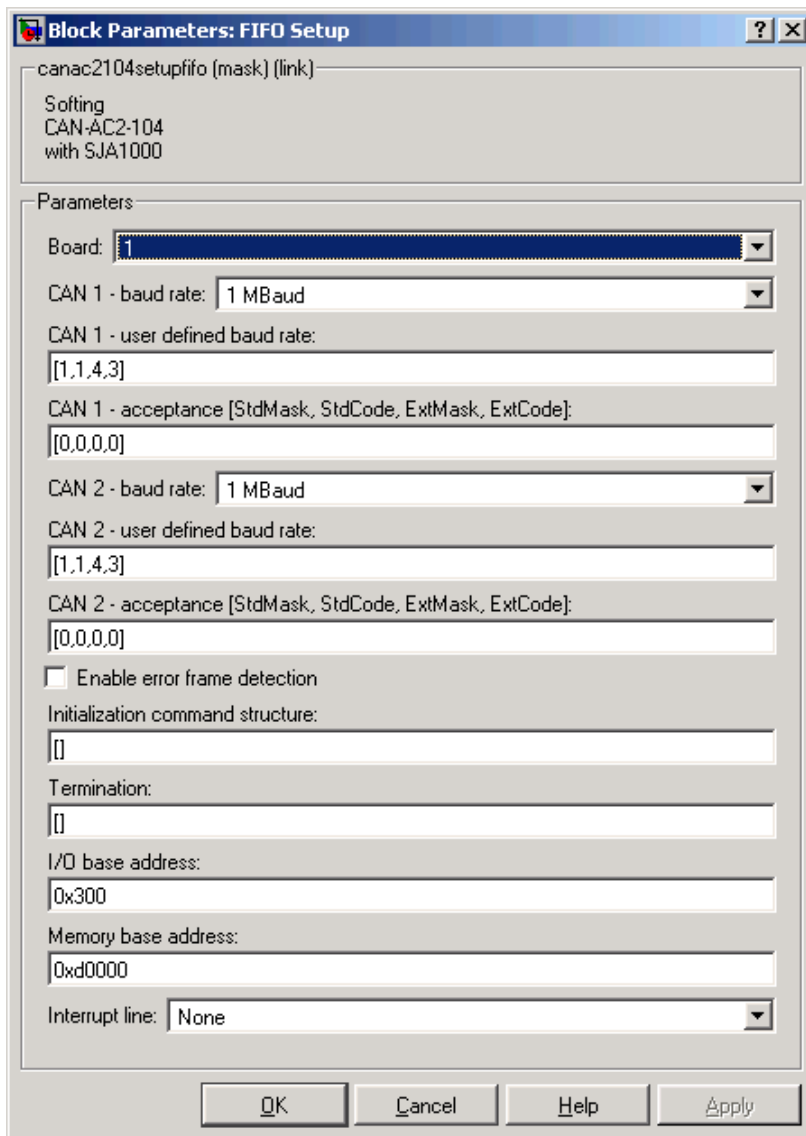
The CAN / IO 308/ FIFO dialog box opens with the driver blocks available for FIFO Mode CAN.



### **FIFO Setup Driver Block (IO 308)**

The FIFO Setup driver block is used to define general settings of the plugged-in CAN boards. The CAN driver blocks for this board support up to three boards for each target system, which leads to the availability of up to six CAN ports. For each board in the target system, you must use exactly one Setup block in a model.

The dialog box of the FIFO Setup block lets you define the following settings.



**Board** — Defines which board is being accessed by this driver block instance. If multiple boards are present in the target system, the board number (1, 2, or 3) can be seen as a reference identifier to differentiate the boards. If just one board is present in the target system, select board number 1.

**CAN 1 - baud rate** — Defines the most common baud rates for CAN port 1. If special timing is necessary (baud rate), you can select `User defined`.

**CAN 1 - user defined baud rate** — If you selected `User defined` from the **CAN 1 - baud rate** list, enter four values for the timing information. The vector elements have the following meanings:

```
[ Prescaler, Synchronization-Jump-Width, Time-Segment-1,  
  Time-Segment-2 ]
```

For more information about these values, see the Softing user manual for this board.

**CAN 1 - acceptance** — Defines the acceptance filters for CAN port 1. Because the receive FIFO is filled with any CAN messages going over the bus, the use of the CAN controller acceptance filters becomes important to filter out unwanted messages already at the controller level. This acceptance filter information is provided by a row vector with four elements, where the first two are used to define the acceptance mask and acceptance code for standard identifiers and the latter two for extended identifiers. The default value defined by the Setup block does not filter out any messages. For information on how to define the acceptance information to filter certain messages, see “CAN I/O Support for FIFO” in xPC Target I/O Reference.

**CAN 2 - baud rate** — Defines the most common baud rates for CAN port 2. If special timing is necessary (baud rate), you can select `User defined`.

**CAN 2- user defined baud rate** — If you select `User defined` from the **CAN 1 - baud rate** list, enter four values for the timing information. The vector elements have the following meanings:

```
[ Prescaler, Synchronization-Jump-Width, Time-Segment-1,  
  Time-Segment-2 ]
```

For more information about these values, see the Softing user manual for this board.

**CAN 2 - acceptance** — Defines the acceptance filters for CAN port 2. Because the receive FIFO is filled with any CAN messages going over the bus, the use of the CAN controller acceptance filters becomes important to filter out unwanted messages already at the controller level. This acceptance filter information is provided by a row vector with four elements, where the first two are used to define the acceptance mask and acceptance code for standard identifiers and the latter two for extended identifiers. The default value defined by the Setup block does not filter out any messages. For information on how to define the acceptance information in order to filter certain messages, see “CAN I/O Support for FIFO” in xPC Target I/O Reference.

**Enable error frame detection** — Defines whether the CAN controller should detect error frames and forward these to the receive FIFO. Checking this box makes sense for monitoring applications where you want to be informed about all events going over the bus. For low latency time applications, checking this box might increase the FIFO Read driver block latency, because the receive FIFO is filled with additional events.

**Initialization command structure and Termination** — Defines CAN messages sent during initialization and termination of the Setup block. For more information, see “CAN I/O Support for FIFO” in the xPC Target I/O Reference.

**I/O base address** — Defines the I/O base address of the board to be accessed by this block instance. The I/O base address is given by the DIP switch setting on the board itself. The I/O address range is 3 bytes and is mainly used to identify which memory base address the board should use. See the Softing user manual for this board on how you can set the I/O base address. The I/O base address entered in this control must correspond with the DIP switch setting on the board. If more than one board is present in the target system, you must enter a different I/O base address for each board. In this case, the I/O base address itself defines which board is referenced by which board number.

**Memory base address** — Defines the memory base address of the board to be accessed by this block instance. The memory base address is a software setting only (no corresponding DIP switch is found on the board). The memory address range is 64 KB. If more than one board is present in the target system, a different memory base address must be entered for each board. You must make sure that the defined address ranges do not overlap. Because the xPC Target kernel only reserves a subset of the address range between 640

KB and 1 MB bytes for memory mapped devices, the address ranges must be within the following range:

C0000-DC000

The board allows activating proper termination for each of the two CAN ports separately by means of DIP switches at the back panel of the board. Refer to the Softing user manual on how to set the DIP switches. Both CAN ports must be properly terminated before you can use the provided model to test the board and drivers.

**Interrupt line** — Select an interrupt line from the list.

### **FIFO Write Driver Block (IO 308)**

The FIFO Write driver block is used to write CAN messages into the transmit FIFO. The firmware running in FIFO mode then processes the information found in the transmit FIFO and finally puts the CAN messages onto the bus.

The block has one input port of type double. At this port all necessary information must be provided to construct valid CAN messages to be written to the transmit FIFO. For each CAN message, you must use five elements:

Port  
Identifier  
Identifier type  
Data frame size  
Data

**Port** — The value can be either 1 (port 1) or 2 (port 2). It defines from which port the CAN message is sent.

**Identifier** — This is the identifier of the CAN message to be sent out if it is a standard CAN message. The valid range is 0 to 2047; if extended, the range is 0 to  $2^{29}-1$ .

**Identifier type** — The value can be either 0 (standard identifier range) or 1 (extended identifier range). It defines the identifier type of the outgoing CAN message.

**Data frame size** — The value can be in the range of 0 to 8. It defines the data frame size of the outgoing CAN message in bytes.

**Data** — This is the data for the data frame itself. It is defined as a double value (8 bytes). The CAN packing block is used to construct the data as a double value.

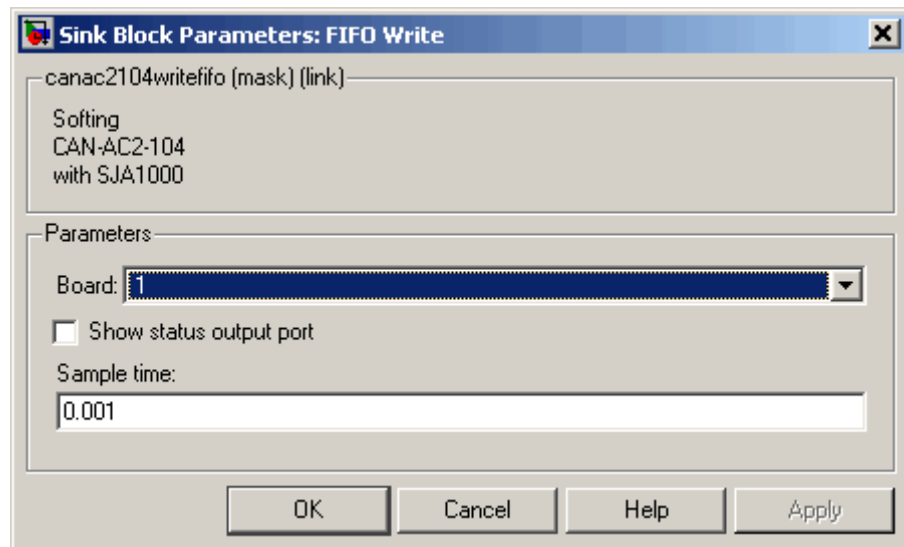
Because all this information can be dynamically changed in FIFO mode during application execution, the information is provided at the block input instead of using the block parameters. To be able to transmit more than one CAN message per block instance, a matrix signal is used as a container for all information.

The dimension of the matrix signal entering the block must be  $n \times 5$ , where  $n$  is the number of CAN messages to be sent by this block instance. Therefore, each row of the matrix signal defines one CAN message and each row combines the five elements of information defined above (in this order).

For more information on how to construct the correct matrix signal for the FIFO write block, see “CAN I/O Support for FIFO” in xPC Target I/O Reference.



For certain applications it might be necessary to make the writing of a CAN message to the transmit FIFO depend on the model dynamics. For this, the matrix signal can also be of dimension  $n*6$  instead of  $n*5$ . In this case, the sixth column defines whether the corresponding CAN message is written to the transmit FIFO (value 1) or not (value 0).



The dialog box of the FIFO Write block lets you define the following settings.

**Board** — Defines which board to use to send the CAN messages defined by this block instance. For more information about the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

**Show status output port** — Selecting this check box lets you enable the status output port. If the box is cleared (disabled), the block does not have an output port. If enabled, a port is shown. The signal leaving the block is a vector of type double, where the number of elements depends on the signal dimension of the block input port. There is one element for each CAN message written to the transmit FIFO and the value is identical to the return argument of function `CANPC_send_data(...)`, described in the Softing user manual. Refer to that manual for more information.

**Sample time** — Defines the sample time at which the FIFO Write block is executed during a model (target application) run.

You can use as many instances of the FIFO Write block in the model as needed. For example, by using two instances of the block, you can define different sample times at which CAN messages are sent. Or you can use multiple instances to structure your model more efficiently.

### **FIFO Read Driver Block (IO 308)**

The FIFO Read driver block is used to read CAN messages from the receive FIFO. The firmware running in FIFO mode puts received events (CAN messages) into the receive FIFO. From here, the FIFO Read driver reads the events out.

The FIFO Read driver block has at least one output port of type double. The signal of this port is a matrix of size  $m \times 6$ , where  $m$  is the FIFO read depth defined in the block's dialog box (see below). For example, if the FIFO read depth is 5, then the matrix signal of port 1 has size  $5 \times 6$ . Therefore there is one row for each event read from the receive FIFO (no new message is considered an event as well). For information on how to extract data from the matrix signal, see “CAN I/O Support for FIFO” in xPC Target I/O Reference.

Each row with its six elements contains all the information defining a CAN message:

- Port
- Identifier
- Event type
- Data frame size
- Timestamp
- Data

**Port** — The value is either 1 (port 1) or 2 (port 2) and reports at which port the CAN message was received.

**Identifier** — Identifier of the CAN message being received. If it is a standard CAN message, the range is 0 to 2047. If the CAN message is extended, the range is 0 to  $2^{29}-1$ .

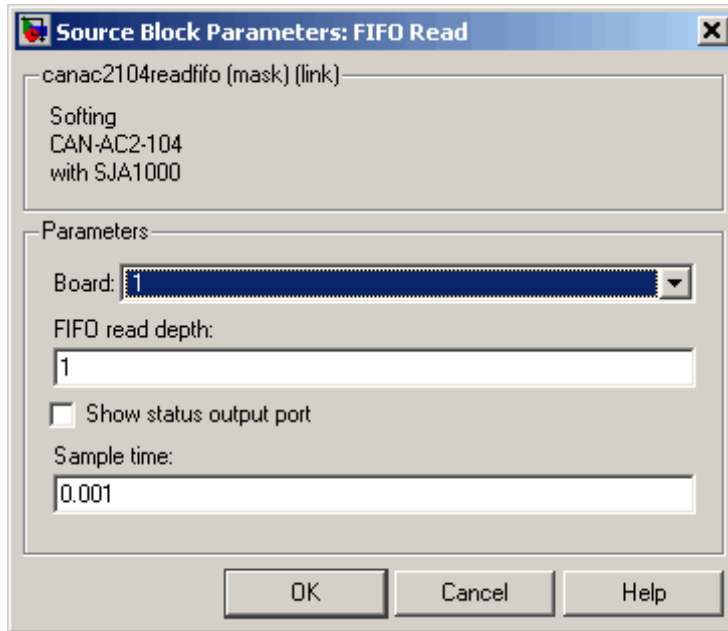
**Event type** — This value defines the type of event read from the receive FIFO. The following values are defined from the Softing user manual:

- 16 No new event
- 17 Standard data frame received
- 18 Standard remote frame received
- 19 Transmission of a standard data frame is confirmed
- 20 -
- 21 Change of bus state
- 22 -
- 23 -
- 24 Transmission of a standard remote frame is confirmed
- 25 Extended data frame received
- 26 Transmission of an extended data frame is confirmed
- 27 Transmission of an extended remote frame is confirmed
- 28 Extended remote frame received
- 29 -
- 30 -
- 31 Error frame detected

**Data frame size** — If a data frame has been received, the length of the data in bytes is reported by this element. Possible values are 0 to 8.

**Timestamp** — This element reports the time at which the event was received. The resolution of the timestamp counter is 1 microsecond.

**Data** — This is the data of the data frame itself. It is returned as a double value (8 bytes). The CAN unpacking block is used to extract the data from the double value.



The dialog box of the FIFO Read block lets you define the following settings.

**Board** — Defines which board to use to send the CAN messages defined by this block instance. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

**FIFO read depth** — Defines the number of receive FIFO read attempts. Each time the block is executed, it reads this fixed amount of events (CAN messages), which leads to a deterministic time behavior independent of the number of events currently stored in the receive FIFO. The Read depth ( $m$ ) defines at the same time the size of the matrix signal ( $m \times 6$ ) leaving the first output port. If no event is currently stored in the receive FIFO, the FIFO is read anyway but the event type is reported as 0 (no new event).

**Show status output port** — Selecting this check box lets you enable the status output port. If the box is cleared (disabled), the block has one output port for the events. If enabled, a second port is shown. The signal leaving that port is a vector of type double with two elements.

[Number of lost messages (events), Bus state]

The first element returns the current value of the lost messages counter. The receive FIFO can store up to 255 events. If the receive FIFO is not regularly accessed for reading events, the FIFO is filled and the lost messages counter starts to increment. This is an indicator that events (messages) will be unavoidably lost. The second element returns the current bus state. Possible values are

- 3 Error active
- 4 Error passive
- 5 Bus off

**Sample time** — Defines the sample time at which the FIFO Read block is executed during a model (target application) run.

---

**Note** Only use one instance of this block per physical CAN board in your model. Otherwise, you might get unwanted behavior where one instance reads events while the events are being processed by blocks connected to the other, second instance.

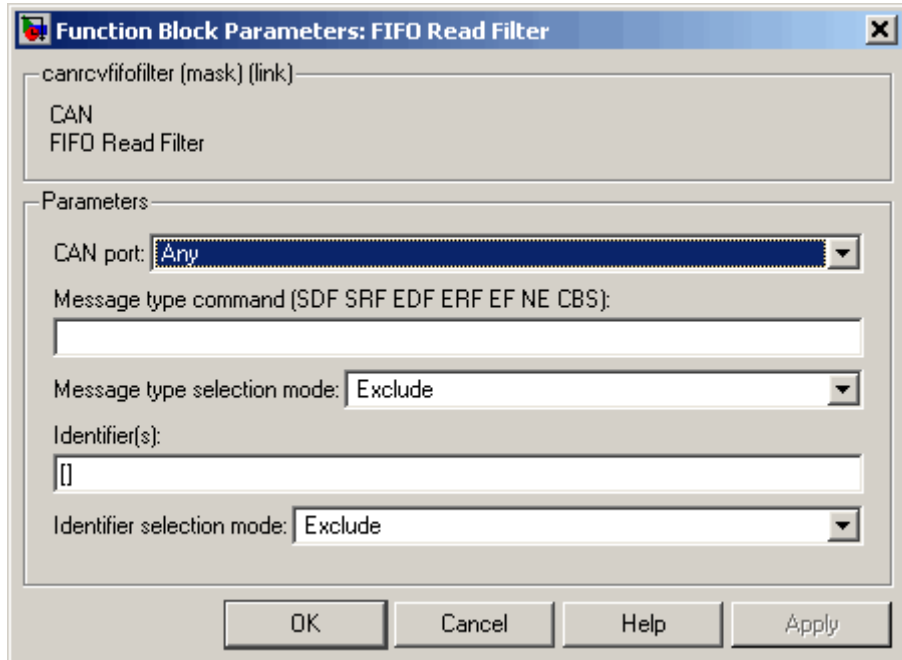
---

## **FIFO Read Filter Block (IO 308)**

This is a utility block for the CAN FIFO driver block set, but does not actually access the CAN board or any other hardware device. This block is usually connected to the first output port of the FIFO Read driver block. It allows filtering events from the event matrix, which is the signal leaving the FIFO Read driver block.

The block code walks through the rows of the incoming event matrix signal and looks for matching events according to the criteria defined in the block's dialog box. If it matches, the entire event information (row) is written to the block's first output port. If more than one row matches the criteria, the later event overwrites the earlier event.

The block has one input port and two output ports. The input port is of type double and accepts a matrix signal of size  $m \times 6$ . The two output ports are of type double as well. The first output is a row vector ( $1 \times 6$ ), the filtered event. The second output is a scalar value that reports the number of matching events the filter block has processed.



The dialog box of the FIFO Read Filter block lets you define the following settings:

**CAN port** — Defines the filter criterion for the CAN port. Possible choices are Any, 1, or 2.

**Message type command** — Defines the filter criterion for the event types. This entry can consist of a concatenation of space-delimited keywords:

SDF Standard data frame  
SRF Standard remote frame  
EDF Extended data frame

ERF Extended remote frame  
EF Error frame  
NE No new event  
CBS Change of bus state

**Message type selection mode** — Defines how the event type (message type) listed in **Message type command** is treated. If you select **Include**, the event type criterion is the sum of the concatenated keywords. If you select **Exclude**, the event type criterion is equal to all event types minus the sum of the concatenated keywords.

**Identifier(s)** — Defines the filter criterion for the CAN message identifiers. A set of identifiers can be provided as a row vector.

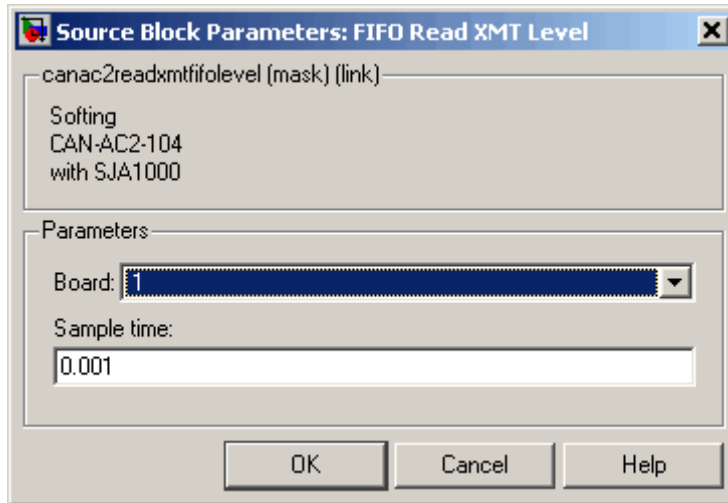
**Identifier selection mode** — Defines how the identifier criterion entered in the control above is treated. If you select **Include**, the identifier criterion is the sum of all specified identifiers. If you select **Exclude**, the identifier criterion is equal to all identifiers minus the specified identifiers.

You can use as many instances of this block in your model as needed. Usually, you connect several instances in parallel to the output of the FIFO Read driver block to filter out particular messages or events. For more information on how to do this, see “CAN I/O Support for FIFO” in xPC Target I/O Reference.

### **FIFO Read XMT Level Driver Block (IO 308)**

The FIFO Read XMT Level driver block is used to read the current number of CAN messages stored in the transmit FIFO to be processed by the firmware. The transmit FIFO can store up to 255 messages. If it is full and a FIFO write driver block tries to add another message to the transmit FIFO, the passed messages are lost. You can use this driver block to check for this condition and take appropriate action. For example, you can stop the execution or wait for the transmit FIFO to empty.

The block has a single output port of type double returning a scalar value containing the current transmit FIFO level (number of messages to be processed).



The dialog box of the FIFO Read XMT block lets you define the following settings:

**Board** — Defines which board to access to read the current transmit FIFO level. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

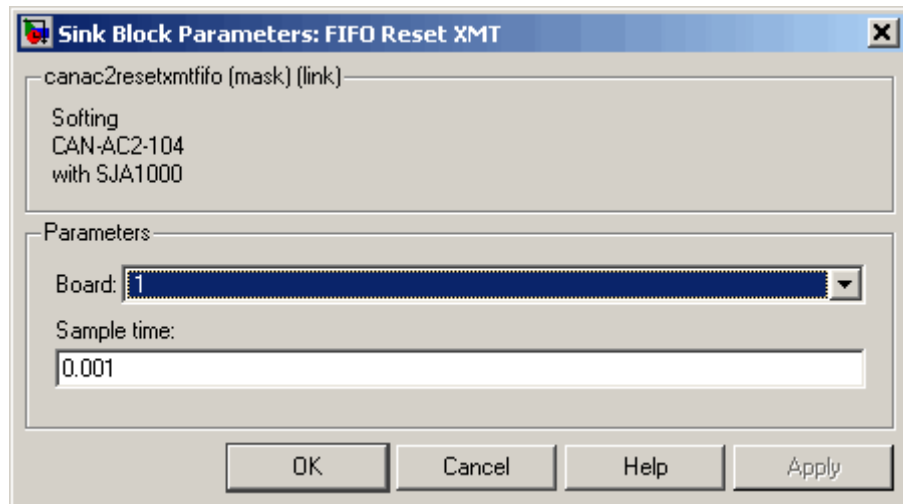
**Sample time** — Defines the sample time at which the FIFO Read XMT Level driver block is executed during a model (target application) run.

### **FIFO Reset XMT Driver Block (IO 308)**

The FIFO Reset XMT driver block is used to reset the transmit FIFO. This deletes all messages currently stored in the transmit FIFO and resets the level counter to 0. For example, you can use this driver block to reset the transmit FIFO after detecting a fault condition.



The block has a single input port of type double. If a scalar value of 1 is passed, the transmit FIFO is reset; if 0 is passed, no action takes place.



The dialog box of the FIFO Reset XMT block lets you define the following settings:

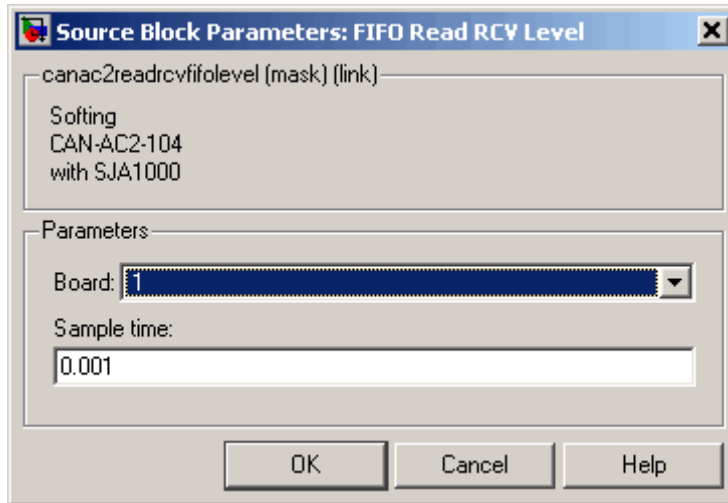
**Board** — Defines which board to access to reset the transmit FIFO. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

**Sample time** — Defines the sample time at which the FIFO Reset XMT driver block is executed during a model (target application) run.

### FIFO Read RCV Level Driver Block (IO 308)

The FIFO Read RCV Level driver block is used to read the current number of CAN messages stored in the receive FIFO. The receive FIFO can store up to 255 events (messages). If it is full and no FIFO Read driver block attempts to read the stored events, new incoming events are lost. This is reflected by the incrementing of the lost message counter. You can use this driver block to check for this condition and take appropriate action (for example, like stopping the execution or resetting the receive FIFO).

The block has a single output port of type double returning a scalar value containing the current receive FIFO level (number of messages to be processed).



The dialog box of the FIFO Read RCV Level block lets you define the following settings:

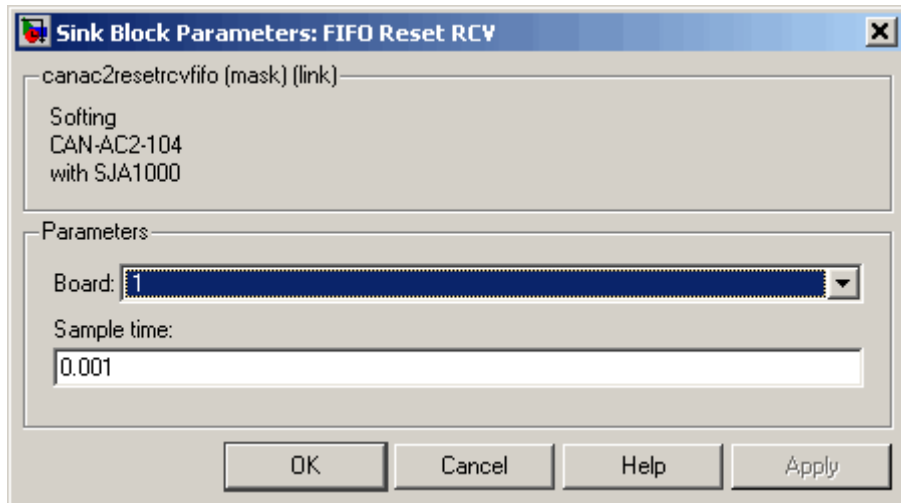
**Board** — Defines which board to access to read the current receive FIFO level. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

**Sample time** — Defines the sample time at which the FIFO Read RCV Level driver block is executed during a model (target application) run.

### **FIFO Reset RCV Driver Block (IO 308)**

The FIFO Reset RCV driver block is used to reset the receive FIFO. This deletes all messages currently stored in the receive FIFO and resets the level counter to 0. As an example, you can use this driver block to reset the receive FIFO after having detected a fault condition.

The block has a single input port of type double. If a scalar value of 1 is passed, the transmit FIFO is reset; if 0 is passed, no action takes place.



The dialog box of the FIFO Reset RCV block lets you define the following settings:

**Board** — Defines which board to access to reset the receive FIFO. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

**Sample time** — Defines the sample time at which the FIFO Reset RCV driver block is executed during a model (target application) run.

## xPC TargetBox IO 309

The MPL PowerPC controlled Analog and Timing I/O Intelligence (PATI) has 32 32-bit TPU channels, which xPC Target supports for PWM measurement, PWM generation, time base selection, incremental encoder measurement, and digital I/O. It is based on the Motorola® MPC555 processor. xPC Target supports only the TPU channels for this board.

This section includes the following topics:

- “Wiring for IO 309 Test Dongle” on page 5-106
- “Pin Layout IO 309” on page 5-106
- “Testing Model IO 309” on page 5-108
- “MPL PATI (IO 309) Driver Blocks” on page 5-109

### Wiring for IO 309 Test Dongle

xPC TargetBox communicates with this board using one connector. Some of the pins in the test dongles are connected together to allow for loopback testing of this board:

- Header (J4) Dongle — The following pins are connected: 1–19 and 2–20.

### Pin Layout IO 309

The following figure is the pin layout for the IO 309 option connector. This is the J4 connector in the manufacturer’s documentation at [http://www.mpl.ch/DOCs/MPLdoc\\_00000178.pdf](http://www.mpl.ch/DOCs/MPLdoc_00000178.pdf). The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.

TPU	CH1	1	2	CH2	TPU		
	CH3	3	4	CH4			
	CH5	5	6	CH6			
	CH7	7	8	CH8			
	CH9	9	10	CH10			
	CH11	11	12	CH12			
	CH13	13	14	CH14			
	CH15	15	16	CH16			
	Gnd	17	18	Gnd			
	CH17	19	20	CH18			
	CH19	21	22	CH20			
	CH21	23	24	CH22			
	CH23	25	26	CH24			
	CH25	27	28	CH26			
	CH27	29	30	CH28			
	CH29	31	32	CH30			
	CH31	33	34	CH32			
	Gnd	35	36	Gnd			
	TPU_T2CLKA	37	38	TPU_T2CLKB			
	SPI	Gnd	39	40		SPI_SDO	SPI
		SPI_SDI	41	42		SPI_CLK	
		SPI_CS0#	43	44		VCC5	
	First and Second RS-232	Gnd	45	46		RXD1	First and Second RS-232
		TXD1	47	48		RXD2	
		TXD2	49	50		Gnd	

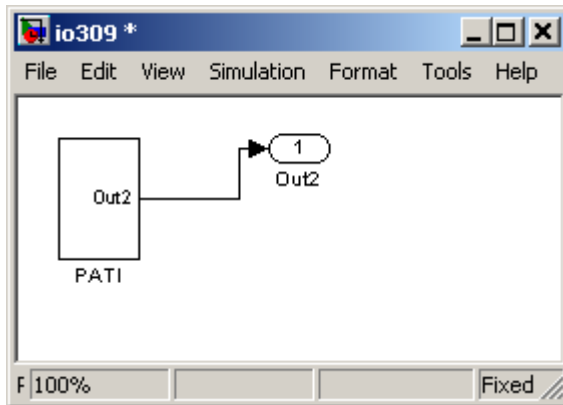
## Testing Model IO 309

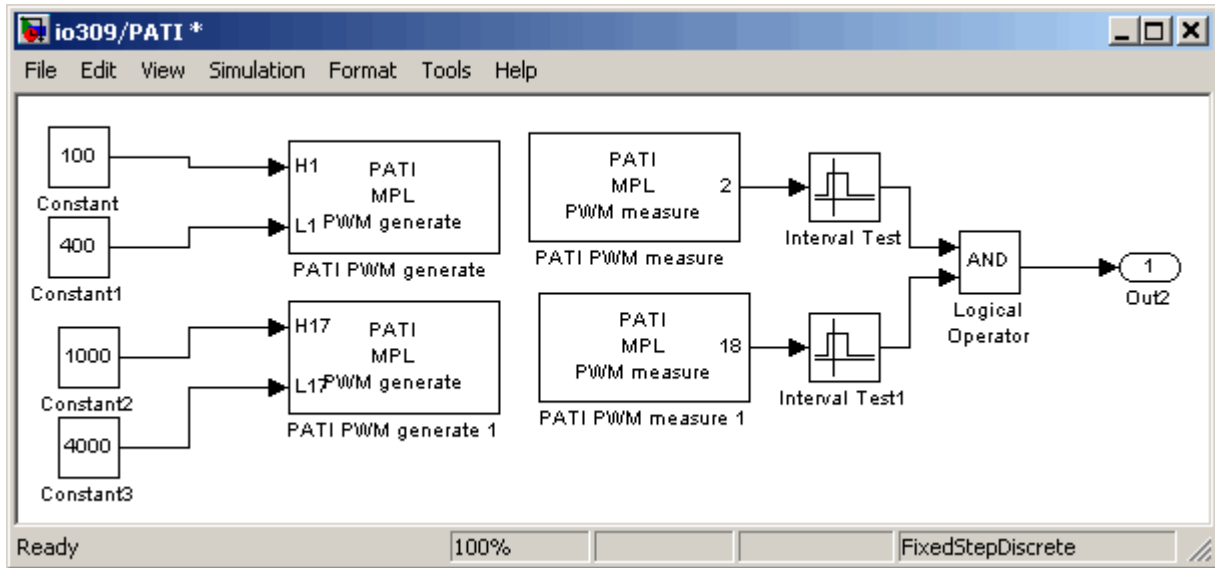
This model tests the PWM generate and PWM measure blocks. Two constant values are written to each of two PWM generate driver blocks and then read from two PWM measure blocks. The values are measured against an error condition defined in an interval test block, and the results are sent to an Output block:

- 1 In the MATLAB Command Window, type

```
io309
```

The Simulink model for testing the IO 309 option opens.





## MPL PATI (IO 309) Driver Blocks

xPC TargetBox supports this board with these driver blocks:

- “MPL PATI Digital Input ” on page 5-110
- “MPL PATI Digital Output” on page 5-111
- “MPL PATI Incremental Encoder” on page 5-112
- “MPL PATI PWM generate” on page 5-114
- “MPL PATI PWM measure” on page 5-117
- “MPL PATI Timebase Setup” on page 5-119

### Board Characteristics

Board name	MPL PATI
Manufacturer	MPL
Bus type	PCI/104
Access method	I/O Mapped

**Board Characteristics (Continued)**

Multiple block instance support	Yes
Multiple board support	Yes

**MPL PATI Digital Input**

**Scaling Input to Output.**

Hardware Input	Block Output Data Type	Scaling
Volts	double	1

**Note.** The individual bits of the MPL PATI digital input are in different registers. As a result, you cannot read multiple bits at the same time. To emphasize this, you must use a separate MPL PATI Digital Input block to read each bit.

**Driver Block Parameters.**

**Channel** — From the list, select from 1 to 32. This is the single TPU channel that this block reads.

**Channel priority** — From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

**Sample time** — Enter the base sample time or a multiple of the base sample time.

**PCI slot** — If only one board of this type is in the target PC, enter

-1

to automatically locate the board.



If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## MPL PATI Digital Output

### Scaling Output to Input.

Hardware Output	Block Input Data Type	Scaling
Volts	double	1

**Note.** The individual bits of the MPL PATI digital output are in different registers. As a result, you cannot update multiple digital outputs at the same time. To emphasize this, you must use a separate MPL PATI Digital Output block to control each bit.

### Driver Block Parameters.

**Channel** — From the list, select from 1 to 32. This is the single TPU channel that this block controls.

**Channel priority** — From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

**Reset** — This parameter controls the behavior of the bit at model termination. Enter one of the following values:

- 1

Resets the output bit to the value in **Initial Value**.

- 0

Leaves output bit at the most recent value attained while the model was running.

**Initial value** — Enter the initial voltage value for the output bit. The output bit is set to this value between the time the model is downloaded and the time the model is started. Also, if the **Reset** parameter has a value of 1, the output bit is reset to the initial value when the model is stopped.

**Sample time** — Enter the base sample time or a multiple of the base sample time.

**PCI slot** — If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## **MPL PATI Incremental Encoder**

**Note.** This block has one input port and one output port.

- **Input port**

Selects the counting mode. The block **M** input port determines the counting mode. The incremental encoder function has two counting modes, normal and fast. In fast mode, the count direction is not controlled by the polarity of the two signals but continues the same direction that was determined when the counter was last in normal mode. The maximum counting rate is much higher in fast mode than in normal mode.

- **Normal mode**

If the block input **M** is less than 0.5, the counter works in normal mode and counts on each edge. The edge sense and polarity of each signal determines the count direction of each signal. Each rising and falling edge of both inputs is counted. You must set the counting mode to normal when the model is first started. You can change it to fast mode afterward.

- Fast mode

If the block input *M* is greater than or equal to 0.5, the counter works in fast mode. The counter increments by 4 each time the primary channel has a rising edge.

- Output port

Returns the current position count as a double.

### Driver Block Parameters.

**Channel pair** — From the list, choose a pair of sequential channels for the incremental encoder. Both channels must be in the same group of 16. For example, 1+2, 2+3, or 18+19. Note that if you want to use channels 16 or 32, their sequential partners are 1 and 17. For example, 16+1 and 32+17.

**Channel priority** — From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

**Initial count** — The initial count specifies the initial value for the encoder count register when the model starts executing.

**Output range** — From the list, choose one of the following. This block converts the output to doubles.

- signed [-32768 +32767]

Interprets the 16-bit count as a signed quantity. This places the discontinuity far from the 0 origin. Counting up from 32767 goes to -32768 on the next count.

- unsigned [0 65535]

Interprets the 16-bit count as an unsigned quantity. This places the discontinuity at the origin, where counting up from 65535 goes to 0 on the next count.

**Sample time** — Enter the base sample time or a multiple of the base sample time.

**PCI slot** — If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

### **MPL PATI PWM generate**

#### **Scaling Output to Input.**

<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
0 to 5 volts	Double	Counts

**Note.** This block uses the PWM module in the time processor units (TPUs) on the board to generate rectangular wave output. See the MPC555 documentation from Freescale Semiconductor at <http://www.freescale.com> for more information on the PWM module.

This block has three inputs:

<b>Input</b>	<b>Description</b>
H	Number of counts of the chosen time base that the output should remain high. You can use a count from 0 to 32767. With the use of multiple channels, you might not be able to smoothly get your output to 0 width. If you have a small number of channels, you can try to use the Channel priority parameter to more smoothly approach 0 width.

Input	Description
L	Number of counts of the chosen time base that the output should remain low. Range from 0 to 32767. With the use of multiple channels, you might not be able to smoothly get your output to 0 width. If you have a small number of channels, you can try to use the Channel priority parameter to more smoothly approach 0 width.
A	Specifies whether or not the output is to be armed. If the A input is 0, the block stops the output and the output level is the one chosen in the dialog. If A is 1, the output runs. This port does not appear if you do not select <b>Arm input</b> .

**Limitations.** If you use multiple blocks for different channels, the relative timing between the output signals is not controlled and is not repeatable on the order of a few 10s of microseconds even if the output frequency is the same.

#### **Driver Block Parameters.**

**Channel** — From the list, select a channel from 1 to 32. This board has two TPUs. Channels 1 through 16 are on the first TPU, channels 17 through 32 are on the second TPU. The channel you select works with the time base parameter.

**Priority** — From the list, select Low , Medium , or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

**Initial high count** — Enter the high count to be used from initialization until the model is running.

**Initial low count** — Enter the low count to be used from initialization until the model is running.

**Immediate update** — Select this check box to force the output to use the new count immediately. Selecting this check box can result in very short output pulses. It terminates long pulses immediately.

Clear this check box to allow the current output cycle to finish before the update. Clearing this check box generates smoother frequency and phase transitions without transients.

The block does not interrupt long pulses. Note that your selection of this parameter value depends on your application.

**Arm input** — Select this check box to indicate that the ARM input is present. If this check box is selected, the channel behaves as follows. Note that there might be a delay, on the order of the output period, between the time that the ARM input becomes 1 and the output port starts changing.

- If the input signal to this port is 1, the channel outputs the signal specified by the H and L ports.
- If the input signal to this port is 0, the output stops at the level specified by the **Disarm state** drop-down list.

If you do not select this check box, the block behaves as if the ARM input is 1. No A input port appears.

**Disarm state** — From the drop-down list, select

- Disarm to LOW
- Disarm to HIGH

When the ARM input is 0, this parameter specifies the state of the output.

**State on model stop** — From the drop-down list, select

- Leave running  
Continue to get output pulses at the last H and L port values.
- Stop to LOW  
Immediately go to the LOW output state and maintain that level.
- Stop to HIGH  
Immediately go to the HIGH output state and maintain that level.

**Timebase** — From the list, select TCR1 or TCR2. This depends on the “MPL PATI Timebase Setup” on page 5-119 settings. If you do not use the MPL PATI Timebase Setup block, the TCR1 time base defaults to 1.25 MHz.

**Sample time** — Enter the base sample time or a multiple of the base sample time.

**PCI slot** — If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**See Also.** “MPL PATI Timebase Setup” on page 5-119

## MPL PATI PWM measure

### Scaling Input to Output.

Hardware Input	Block Output Data Type	Scaling
0 to 5 volts	Double	Counts of current time base

**Note.** This block uses the PTA module in the time processor units (TPUs) on the board to measure full cycle period or high or low time in an input waveform. To properly use the TPUs, you must correctly condition the TPU inputs with a voltage between 0 and 5 volts. The inputs also must have a reasonably fast rise or fall time; otherwise, false triggering might occur. The inputs use TTL level transitions. See the MPC555 documentation from Freescale Semiconductor (<http://www.freescale.com>) for more information on the PTA module. This block outputs the number of counts of the input time base (default 20 MHz. configurable in the “MPL PATI Timebase Setup” on page 5-119 block) that meet the mode chosen.

### **Driver Block Parameters.**

**Channel** — From the list, select a channel from 1 to 32. This board has two TPUs. Channels 1 through 16 are on the first TPU, channels 17 through 32 are on the second TPU. The channel you select works with the time base parameter.

**Channel priority** — From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

**Mode** — From the list, select

- Total High Time — For measuring pulse widths
- Total Low Time — For measuring pulse widths
- Between Rising Edges — For period measurements
- Between Falling Edges — For period measurements

**Number of periods to count** — Enter the number of periods to count. To measure every cycle separately, enter a value of 1 (default). To get higher precision by averaging, enter a higher value, then divide the block output by this value. For example, to gain a period that is accurate to 0.1 count, enter a value of 10 for this count, then divide the block output by 10.

**Timebase** — From the list, select TCR1 or TCR2. This depends on the “MPL PATI Timebase Setup” on page 5-119 settings. If you do not use the MPL PATI Timebase Setup block, the TCR1 time base defaults to 1.25 MHz.



**Sample time** — Enter the base sample time or a multiple of the base sample time.

**PCI slot** — If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**See Also.** “MPL PATI Timebase Setup” on page 5-119

## MPL PATI Timebase Setup

**Note.** The Motorola MPC555 processor contains two time processor units (TPUs). Each TPU has 16 timer channels (1 to 16 and 17 to 32), totaling 32 channels, for one MPL PATI board. Each TPU has two time bases (four between the two TPUs) that can be used in the timer functions. The basic frequency for the TPU is the clock frequency for the CPU (40 MHz). The minimum divider that can be applied is a factor of 2, which gives the maximum timer frequency of 20 MHz. This block allows you to set the frequency for each of the four time bases, two time bases for channels 1 to 16 and two for channels 17 to 32.

Each TPU allows two different timer sources called TCR1 and TCR2.

- TCR1 is driven by a clock source with a frequency of  $40 \text{ MHz}/N$ , where  $N$  is a power of 2 between 2 and 512.
- TCR2 can be driven either by an external clock source or by a clock source with a frequency of  $40 \text{ MHz}/N$ , where  $N$  is a power of 2 between 8 and 32. Note that the MPL PATI user documentation refers to the channel's 17–32 external input as TPU\_T2CLKB.

The hardware architecture restricts allowable values for the time bases. See the MPC555 documentation from Freescale Semiconductor at <http://www.freescale.com> for more information on the TPUs and time bases.

### **Driver Block Parameters.**

**TCR1 divider (1–16)** — From the list, choose the divider that sets the TCR1 time base for channels 1 to 16. The minimum divider of 2 gives the maximum time base frequency of 20 MHz. The maximum divider of 512 gives a time base frequency of 78.125 KHz.

**TCR2 divider (1–16)** — From the list, choose one of the following:

- 8 (5 MHz) or ext
- 16 (2.5 MHz) or ext/2
- 32 (1.25 MHz) or ext/4
- 64 (625 KHz) or ext/8

This divider sets the TCR2 time base for channels 1 to 16. You can specify that the TCR2 time base can be driven by either the 40 MHz CPU clock or by a signal on the external T2CLK input pin. Note that the maximum frequency of the external signal is approximately  $40 \text{ MHz}/9 = 4.44 \text{ MHz}$ .

If you want to use the external clock, select the **TCR2 external clock input (1–16)** check box.

**TCR2 external clock input (1–16)** — Select this check box to use the signal on the T2CLK pin as the source for TCR2. Use this parameter with the **TCR2 divider (1–16)** parameter.

**TCR1 divider (17–32)** — From the list, choose the divider that sets the TCR1 time base for the channels 17 to 32. A minimum divider of 2 gives the maximum time base frequency of 20 MHz. A maximum divider of 512 gives a time base frequency of 78.125 KHz.

**TCR2 divider (17–32)** — From the list, choose one of the following:

- 8 (5 MHz) or ext
- 16 (2.5 MHz) or ext/2
- 32 (1.25 MHz) or ext/4
- 64 (625 KHz) or ext/8

This divider sets the TCR2 time base for channels 1 to 16. You can specify that the TCR2 time base can be driven by either the 40 MHz CPU clock or by a signal on the external T2CLK input pin. Note that the maximum frequency of the external signal is approximately  $40 \text{ MHz}/9 = 4.44 \text{ MHz}$ .

If you want to use the external clock, select the **TCR2 external clock input (17–32)** check box.

**TCR2 external clock (17–32)** — Select this check box to use the signal on the T2CLK pin as the source for TCR2. Use this parameter with the **TCR2 divider (17–32)** parameter.

**PCI slot** — If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



## A

- analog input
  - I/O option 301 1-19
- analog output
  - I/O option 301 1-19
  - I/O option 302 1-19
  - I/O option 303 1-20
- application
  - downloading loopback test 3-31
  - with DOSLoader mode 4-10
  - with StandAlone mode 4-13

## B

- BIOS
  - QuickBOOT capability 1-8
- BootFloppy mode 1-9
- booting
  - DOS instead of application 4-22
  - FTP server 4-17

## C

- C compiler
  - setup dialog box 3-14
- CAN controller
  - I/O option 308 1-21
  - onboard 1-12
- commands
  - FTP server 4-18
- communication
  - network 1-10
  - serial 1-10
  - standard PC peripherals 1-10
  - with host PC 1-10
  - xPC TargetBox connectors 1-16
- configuration label
  - I/O options 2-19
- connecting
  - external floppy disk drive 3-8

- peripherals 3-11
- screw terminal boards 2-28

## connectors

- pin layout I/O option 301 analog 5-11
- pin layout I/O option 301 digital 5-13
- pin layout I/O option 302 5-27
- pin layout I/O option 303 5-35
- pin layout I/O option 304 5-43
- pin layout I/O option 305 5-50
- pin layout I/O option 306 5-64
- pin layout I/O option 308 5-74
- pin layout I/O option 309 5-106
- xPC TargetBox 5-4

## contacting The MathWorks

- for technical support 3-32

## counters

- I/O option 305 1-20

## CPU

- xPC TargetBox 1-16

## creating

- application with DOSLoader mode 4-10
- application with StandAlone mode 4-13
- test model 5-7

## D

### digital I/O

- I/O option 301 1-19
- I/O option 302 1-19
- I/O option 304 1-20

### directories and files

- on flash disk 4-15

### dongles

- I/O option 301 wiring 5-10
- I/O option 302 wiring 5-27
- I/O option 303 wiring 5-35
- I/O option 304 wiring 5-43
- I/O option 305 wiring 5-50
- I/O option 306 wiring 5-64
- I/O option 308 wiring 5-74

- I/O option 309 wiring 5-106
- removing 3-7
- DOS
  - on host PC with FTP 4-18
- DOSLoader mode
  - copying kernel 4-8
  - creating target application 4-10
  - feature 1-9
- driver support
  - I/O options library 4-2
  - introduction to I/O options 1-11
  - LED 4-4
  - watchdog 4-5

## **E**

- Embedded Option
  - DOSLoader mode 1-9
  - StandAlone mode 1-9
- encoders
  - I/O option 306 1-21
- environment
  - network communication 3-17
  - serial communication 3-14
- environment properties
  - and StandAlone mode 4-12
  - network communication 3-17
  - serial communication 3-14
- external connectors
  - system hardware 1-16
  - with I/O options 1-11
- external floppy disk drive
  - connecting 3-8
  - xPC TargetBox 1-16

## **F**

- files
  - on flash disk 4-15
- flash disk 1-7

- directories 4-15
- nonvolatile memory 1-16
- flash memory 1-7
- FlashRAM 1-7
- floppy disk drive
  - connecting 3-8
- FreeDOS
  - copying kernel 4-8
  - included with xPC TargetBox 1-8
- FTP server
  - booting 4-17
  - commands 4-18
  - included with xPC TargetBox 1-8
  - with DOS shell 4-18

## **H**

- hardware
  - connecting 2-18
  - network communication 3-17
  - planning connections 2-27
  - requirement, host PC 1-15
  - screw terminal boards 2-28
  - xPC TargetBox features 1-7
  - xPC TargetBox summary 1-16
- host PC
  - hardware 3-14
  - hardware requirements 1-15
  - software requirements 1-14
- Host PCs
  - configuring 3-4

## **I**

- I/O board options 5-1
- I/O expansion slots 1-16
- I/O option 301
  - connector pin layout, analog 5-11
  - connector pin layout, digital 5-13
  - wiring test dongles 5-10

- I/O option 302
    - connector pin layout 5-27
    - wiring test dongles 5-27
  - I/O option 303
    - connector pin layout 5-35
    - wiring test dongles 5-35
  - I/O option 304
    - connector pin layout 5-43
    - wiring test dongles 5-43
  - I/O option 305
    - connector pin layout 5-50
    - wiring test dongles 5-50
  - I/O option 306
    - connector pin layout 5-64
    - wiring test dongles 5-64
  - I/O option 308
    - connector pin layout 5-74
    - wiring test dongles 5-74
  - I/O option 309
    - connector pin layout 5-106
    - wiring test dongles 5-106
  - I/O options
    - configuration label 2-19
    - description 1-18
    - driver library 4-2
    - driver support 1-11
    - expandability 1-11
    - expansion slots 1-11
    - external connectors 1-11
    - features 1-11
    - IO 301 A/D, D/A DIO 1-19
    - IO 302 D/A 1-19
    - IO 303 D/A 1-20
    - IO 304 DIO 1-20
    - IO 305 counters 1-20
    - IO 306 encoders 1-21
    - IO 308 CAN field bus 1-21
    - IO 309 MPL PATI 1-21
    - pin layout 5-3
  - installation
    - testing 3-27
    - unpacking shipping case 2-2
  - installing
    - hardware 3-14
    - xPC Target Embedded Option 3-6
  - installing xPC TargetBox
    - additional peripherals 3-11
    - external floppy disk drive 3-8
- J**
- jumper codes
    - I/O options 2-19
- K**
- kernel
    - copying to flash memory 4-8
    - with DOSLoader mode 4-8
    - with StandAlone mode 4-13
- L**
- LED
    - driver blocks 4-4
  - loopback test
    - troubleshooting 2-15
- M**
- MPL PATI
    - I/O option 309 1-21
- N**
- network communication
    - environment properties 3-17
    - feature 1-10
    - hardware 3-17
    - host PC 3-17
    - setting up 3-17

target PC 3-17

## **P**

PC compatibility 1-6

peripherals

connecting 3-11

xPC TargetBox 1-16

pin layouts

I/O option 301 analog 5-11

I/O option 301 digital 5-13

I/O option 302 5-27

I/O option 303 5-35

I/O option 304 5-43

I/O option 305 5-50

I/O option 306 5-64

I/O option 308 5-74

I/O option 309 5-106

I/O options 5-3

terminal boards 5-3

pin numbering

connectors 5-11

screw terminal boards 5-11

planning

hardware connections 2-27

## **Q**

QuickBOOT capability 1-8

## **R**

RAM

xPC TargetBox 1-16

reboot test 3-31

## **S**

screw terminal boards

connecting 2-28

pin layout 5-3

self-test

on flash disk 1-15

running 2-11

running with monitor 2-11

running without monitor 2-6

troubleshooting 2-15

serial communication

environment properties 3-14

feature 1-10

hardware 3-14

setting up 3-14

setting parameters

network communication 3-17

serial communication 3-14

setup dialog box

C compiler 3-14

shipping case

picture 2-2

unpacking 2-2

Simulink model

test 5-7

test I/O option 302 5-29

test I/O option 303 5-37

test I/O option 304 5-45

test I/O option 305 5-52

test I/O option 306 5-68

test I/O option 308 5-75

test I/O option 309 5-108

software

FreeDOS 1-8

FTP server 1-8

host PC 1-14

QuickBOOT 1-8

self-test 1-8

xPC TargetBox 1-8

StandAlone mode

creating kernel/application 4-13

updating environment properties 4-12

with Embedded Option 1-10

standard PC peripherals 1-10



- system memory 1-7
  - system ping
    - troubleshooting 3-29
  - system requirements
    - hardware, host PC 1-15
    - software, host PC 1-14
- T**
- target application
    - downloading loopback test 3-31
    - running test 5-8
    - with DOSLoader mode 4-10
  - target boot disk
    - creating with GUI 3-22
  - target PC
    - hardware 3-14
  - terminal boards
    - making your own 5-4
    - numbering 5-3
  - test
    - determining success 5-6
    - running target application 5-8
    - Simulink model 5-7
    - Simulink model I/O option 302 5-29
    - Simulink model I/O option 303 5-37
    - Simulink model I/O option 304 5-45
    - Simulink model I/O option 305 5-52
    - Simulink model I/O option 306 5-68
    - Simulink model I/O option 308 5-75
    - Simulink model I/O option 309 5-108
    - testing process 5-5
    - uses for 5-6
  - testing
    - I/O option 302 5-29
    - I/O option 303 5-37
    - I/O option 304 5-45
    - I/O option 305 5-52
    - I/O option 306 5-68
    - I/O option 308 5-75
    - I/O option 309 5-108
  - troubleshooting
    - downloading application test 3-31
    - reboot test 3-31
    - self-test 2-15
    - system ping 3-29
    - xPC Target ping 3-30
- W**
- watchdog, driver block 4-5
- X**
- xPC Target
    - communication ping 3-30
    - installing 3-3
  - xPC Target Embedded Option
    - and host PC 1-13
    - booting without 1-9
    - installing 3-6
  - xPC TargetBox
    - and xPC Target 1-14
    - BIOS 1-15
    - connecting hardware 2-18
    - CPU 1-16
    - description of I/O options 1-18
    - external AC adapter 1-7
    - external connectors 1-16
    - external floppy disk drive 1-16
    - external power supply 1-16
    - files and directories 4-15
    - FreeDOS 1-15
    - FTP server 1-15
    - hardware 1-16
    - hardware summary 1-7
    - LED driver block 4-4
    - library drivers 4-2

- PC compatibility 1-6
- peripherals 1-16
- powering up 3-24
- RAM 1-16
- software 1-15
- software summary 1-8
- temperature range 1-7
- watchdog driver block 4-5
- xPC TargetBox IO 301 5-10
- xPC TargetBox IO 302 5-27
- xPC TargetBox IO 303 5-35
- xPC TargetBox IO 304 5-43
- xPC TargetBox IO 305 5-50
- xPC TargetBox IO 306 5-64
- xPC TargetBox IO 308 5-74
- xPC TargetBox IO 309 5-106
- xPC TargetBox model 106 1-17
- xPC TargetBox model 107 1-17
- xPC TargetBox model 108 1-17
- xPC TargetBox model 206 1-18
- xPC TargetBox model 207 1-18
- xPC TargetBox model 208 1-18