xPC TargetBox™

For Use with Real-Time Workshop®

- Modeling
- Simulation
- Implementation

User's Guide

Version 2

How to Contact The MathWorks:

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

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

508-647-7000 Phone

508-647-7001 Fax

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

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

xPC TargetBox User's Guide

Printing History: November 2002 First printing New for Version 2 (Release 13)
September 2003 Second printing Revised for Version 2.0.1 (Release 13 SP1)
October 2004 Online only Revised for Version 2.0.3 (Release 13SP2)

Contents

Preface

1 Introduction

2 Initial Loop-Back Test

3 Regular Use

4 Advanced Use

5 xPC TargetBox I/O Options

Index

Preface

xPC TargetBox™ is a hardware product you use with xPC Target software to build, test, validate, and deploy real-time systems.

What Is xPC TargetBox?

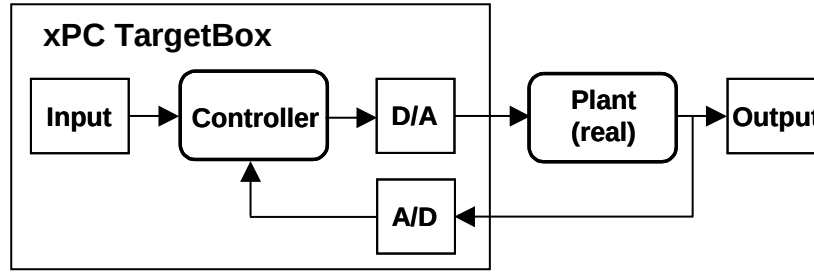
xPC Target provides the capability for prototyping, testing, and deploying real-time systems using standard PC hardware. It is an environment that uses a target PC, separate from a host PC, for running real-time applications.

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.

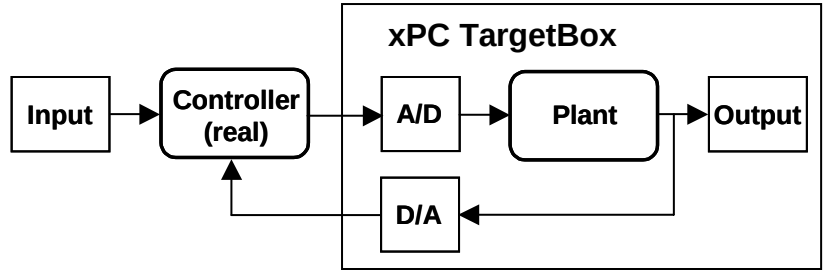
Support for Development Process

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 realtime 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 the 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 box 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 Box” on page 2-2.

2. Remove 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.

3. Attach the AC adapter and the AC power cord to 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 2-4.

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 2-9.

6 Set up xPC TargetBox for use with xPC Target.
   - See “xPC Target Software Installation” on page 3-2.
   - See “xPC TargetBox Hardware Installation” on page 3-6.
   - See “Host PC to xPC TargetBox Communication” on page 3-12.

7 Create an xPC Target boot disk for `BootFloppy` mode.

   See “Creating a Target Boot Disk” on page 3-18.

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

   See “Booting an xPC TargetBox” on page 3-20.

Using This Guide

Reading a brief description of the chapters in this book and a suggested reading path will help you use this guide effectively. This section includes the following topics:

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 Loop-Back Test," then complete the installation following procedures in Chapter 3, "Regular Use."

Continue by reading Chapter 1, "Introduction," and the introduction in the xPC Target Getting Started 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 documentation 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.

How This Book Is Organized

This book explains how to set up a host PC/xPC TargetBox computer environment. It also explains the basic tasks to create and run a target application on an xPC TargetBox. The following table lists the organization of the xPC TargetBox User's Guide.

Chapter	Description
Preface	Brief description of an xPC TargetBox and information about this book
Chapter 1, "Introduction"	Overview of the functions and features of xPC TargetBox
Chapter 2, "Initial Loop-Back Test"	Testing a new xPC TargetBox with a preinstalled custom target application
Chapter 3, "Regular Use"	Connecting an xPC TargetBox to a host computer, booting from a floppy disk drive, and testing a demo application
Chapter 4, "Advanced Use"	Using the DOSLoader and StandAlone boot modes with an xPC TargetBox, and transferring files from a host PC to an xPC TargetBox with the FTP server
Chapter 5, "xPC TargetBox I/O Options"	Reference information for the xPC TargetBox I/O options and a description of the loop-back testing procedures

Warranty and Repair for an xPC TargetBox

xPC TargetBox comes with a one-year limited warranty provided by MPL (xPC TargetBox is designed by MPL for The MathWorks exclusively). During this time, MPL will repair defects in material and workmanship under the terms of the MPL warranty. This section includes the following topics:

- "Limited Warranty from MPL" on page xiv — Statement of warranty, exclusive remedy, and limitation of liability.
- "Repairing an xPC TargetBox Under Warranty" on page xvi — Contact The MathWorks for instructions to ship your xPC TargetBox directly to MPL.
- "Contacting The MathWorks for Technical Support" on page xvi — Use e-mail, the Internet, or telephone to get help with your problem.
- "xPC TargetBox Certification" on page xvii — The xPC TargetBox has been certified by passing emission tests required for use in the US, Europe, Japan, Canada, and Australia/New Zealand.

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 "Contacting The MathWorks for Technical Support" on page xvi 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 Box" on page 2-2.

3. MPL will repair your xPC TargetBox and return it to you within the shortest possible time.

Contacting The MathWorks for Technical Support

If you are having a problem with your xPC TargetBox that you cannot solve, contact The MathWorks directly for help.

Internet	`http://www.mathworks.com/support/`
E-mail	`mailto:support@mathworks.com`
Telephone	508-647-7000 Ask for Technical Support.

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)

1

Introduction

An understanding of the features of xPC TargetBox and the xPC Target software environment will help you develop a process for working with these products. This chapter includes the following sections:

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" on page 1-2 — Description of the hardware and external connections
- "Software" on page 1-4 — Description of the BIOS, DOS, FTP server, and a custom target application for system testing preinstalled on the flash disk
- "xPC TargetBox Boot Modes" on page 1-5 — Boot the kernel from the flash disk and download a target application from a host PC, or create a stand-alone target application and boot it from the flash disk
- "Communication" on page 1-6 — Communicate with an xPC TargetBox through standard peripherals, serial ports, and a network connection
- "I/O Options" on page 1-6 — To an xPC TargetBox, add options that satisfy the needs of most xPC Target applications

Hardware

The xPC TargetBox 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 flash disk.

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

- 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 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.
- **(DOSLoader)/StandAlone mode** — Copy the xPC Target kernel with a self-contained target application, using a floppy disk or the FTP server. After transferring the files, you can remove the floppy disk drive and 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 Boot Modes” on page 3-3.

xPC Target Embedded Option

The xPC Target Embedded Option is an extension to xPC Target. To use the API or COM API interface from the Embedded Option with an xPC TargetBox, you need to purchase a license for the xPC TargetBox Embedded Option.

You also need to purchase a license for the xPC Target Embedded Option to use the DOSLoader and StandAlone boot modes, API, or COM API on a target PC system that is not an xPC TargetBox.

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

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.

For more information about the available I/O options, see “xPC TargetBox I/O Options” on page 1-14.

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.

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" on page 1-8 — MATLAB, Simulink, Real-Time Workshop, xPC Target, and C/C++ compiler
- "Hardware Requirements for a Host PC" on page 1-10 — Desktop or notebook computer
- "xPC TargetBox Software" on page 1-10 — FreeDOS, QuickBoot capability for Pentium III based systems, and a custom target application for testing your system
- "xPC TargetBox Hardware" on page 1-12 — Pentium CPU, flash disk, standard PC connections, slots for three I/O boards, and an external floppy disk drive
- "xPC TargetBox I/O Options" on page 1-14 — 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 6.5.2 (Release 13 SP2)
Simulink	Version 5.2 (Release 13 SP2). Create physical models with block diagrams.
Stateflow (optional)	Version 5.1 (Release 13 SP2). Add state charts to a Simulink model.

Software	Description
Real-Time Workshop	Version 5.2 (Release 13 SP2). Generate code from Simulink models.
Stateflow Coder (optional)	Version 5.1 (Release 13 SP2). Use with Real-Time Workshop to generate code from Simulink models with state charts.
C programming language compiler	Microsoft Visual C/C++ Version 5.0, 6.0, or 7.0, or Watcom C/C++ Version 10.6 or 11.0. Use with Real-Time Workshop and xPC Target to generate executable code to run with the xPC Target kernel.
xPC Target	Version 2.0.3 (Release 13 SP2) 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 2.0.3 (Release 13 SP2). 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.

Software	Description
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 Reference Guide documentation provided in the shipping case from MPL. The following table summarizes the hardware.

Hardware	Description
Communication	Serial ports (COM1, COM2) 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.
External I/O connectors	Front and back panels with shielded I/O connectors for I/O board options Onboard CAN bus connector (not currently supported). Use IO option 308 for a CAN bus interface. Connectors for standard PC peripherals (monitor, mouse, keyboard, four RS-232 ports, one parallel port, and RJ-45 Ethernet)

Hardware	Description
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.

Extended Temperature Unit Name	Description
xPC TargetBox 206	266 MHz Pentium MMX, 128 MB RAM, 32 MB FlashRAM. Extended temperature from -40°C to 75°C.

Extended Temperature Unit Name	Description
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 they are connected to.

I/O Option Name	Description
xPC TargetBox IO 301 Diamond-MM-32-AT Extended temperature -40°C to 85°C	**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 μs). Channel coupling selected with jumpers on board. If you do not specify a channel coupling when purchasing an xPC TargetBox, the coupling is set to 16 differential channels. Unipolar and bipolar input ranges of 0-10 V, 0-5 V, 0-2.5 V, 0-1.25 V, 0-0.625 V, ±10 V, ±5 V, ±2.5 V, ±1.25 V, and ±0.625 V selected by software.
	Analog output (D/A) — 4 (12-bit) channels with a 6 μs settling time. Output range set to 0-10 V, 0-5 V, ±10 V, or ±5 V with jumpers on the board. If you do not specify an output range when purchasing an xPC TargetBox, the range is set to ±10 V.
	Digital I/O — 24 digital I/O lines divided into three groups with 8 bits each. xPC Target does not support the counters on this board.

I/O Option Name	Description (Continued)
xPC TargetBox IO 302 Diamond Ruby-MM-1612 Extended temperature -40°C to 85°C	**Analog output (D/A)** — 16 (12-bit) channels with a 6 μs settling time. Output range for each group of 8 channels set to 0-10 V, 0-5 V, 0-2.5 V, ±10 V, ±5 V, or ±2.5 V with jumpers on the board. If you do not specify a range when purchasing an xPC TargetBox, the range is set to ±10 V. **Digital I/O** — 24 TTL digital I/O lines divided into three groups with 8 bits each xPC Target does not support the external trigger on this board.
xPC TargetBox IO 303 Diamond Ruby-MM-416 Standard temperature 0°C to 70°C	**Analog output (D/A)** — 4 (16-bit) channels with a 10 μs settling time. Output range for each channel set independently to 0-10 V, ±10 V, or ±5 V with jumpers on the board. If you do not specify a range when purchasing an xPC TargetBox, the range is set to ±10 V. **Digital I/O** — 24 TTL digital I/O lines divided into three groups with 8 bits each. xPC Target does not support the external trigger on this board.
xPC TargetBox IO 304 Diamond Onyx-MM Extended temperature -40°C to 85°C	**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. xPC TargetBox does not support the counter/timers and external interrupts on this board.

I/O Option Name	Description (Continued)
xPC TargetBox IO 305 Diamond Quartz-MM-10 Standard temperature 0°C to 60°C	**Counters** — 10 (16-bit) general purpose counters for pulse-train generation and pulse-width measurements. Maximum clock frequency 4 MHz. **Digital I/O** — 8 TTL digital input lines and 8 TTL digital output lines. **Interrupt** — 1 digital interrupt line.
xPC TargetBox IO 306 Real Time Devices DM6814 Standard temperature 0°C to 70°C	**Encoders** — 3 up/down (16-bit) counters with a maximum input rate of 1 MHz. **Digital I/O** — 18 TTL digital input-only lines and 6 TTL input or output lines. **Interrupts** — 2 digital interrupt lines. xPC TargetBox does not support the counter/timers on this board.
xPC TargetBox IO 308 Softing CAN-AC2-104 Extended temperature -40°C to 85°C	**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). 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.

2

Initial Loop-Back Test

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 loop-back the output signals to inputs. This chapter includes the following sections:

Installation for Loop-Back Testing

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

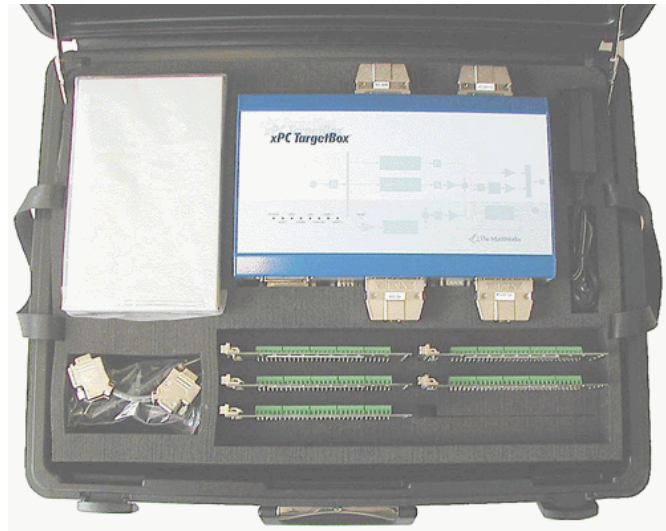
- "Unpacking the Shipping Box" on page 2-2 — Remove the xPC TargetBox from the plastic shipping case
- "Running the Self-Test Without a Monitor" on page 2-4 — Connect the AC power adaptor and run the self-test
- "Running the Self-Test With a Monitor" on page 2-9 — Connect a monitor to the xPC TargetBox and use the self-test to isolate problems
- "Troubleshooting the Loop-Back Self-Test" on page 2-12 —Use information from the self-test to solve problems identified with this test

Unpacking the Shipping Box

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 Place the shipping case on a flat surface so you can read the labels on either side of the shipping case handle.

2 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.

3 Remove the external floppy disk drive to uncover the drive cable.

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

4 Remove the upper layer by its handles.

Contents of the lower layer — External I/O cables, *xPC TargetBox User's Guide*, MPL hardware reference manual, and I/O reference manuals.

5 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 upper layer, remove the xPC TargetBox. Check visually for any physical damage.

2 Turn the box over, and on the bottom, find the configuration label. 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.

3 Turn the box 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 and attach it to the CAN 1 and CAN 2 connectors.

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.

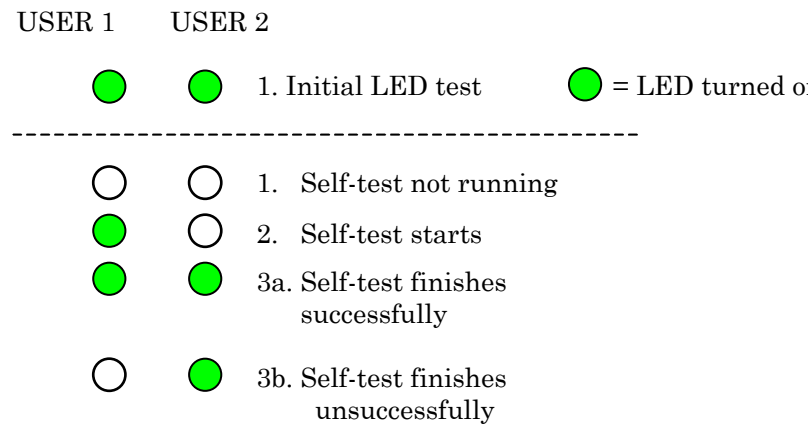
7 From the top compartment, remove the AC power cord.

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 box. 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 2-9.

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 2-4. The AC power cord should be unplugged.

   Find a monitor with a resolution of at least 640x480 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.

   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 box and the monitor screen.

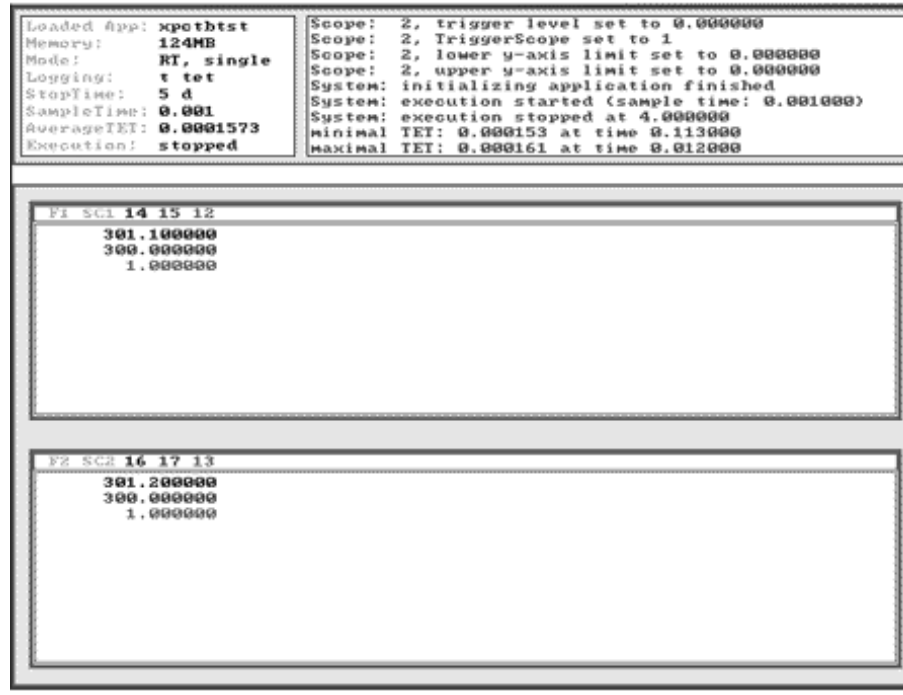
The sequence of LEDs turning on and off should follow the sequence described in “Running the Self-Test Without a Monitor” on page 2-4. You should see a display on the monitor for the first time. The initial content you see on the screen depends on the model of your xPC TargetBox.

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.



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. For example, 301.1 is the analog connector for the IO 301 option, and 301.2 is the digital connector for the same I/O option. The next number is a base address (in this example, 300). This is the base address for the board for which the test application was created. Below the base address is a numerical value.

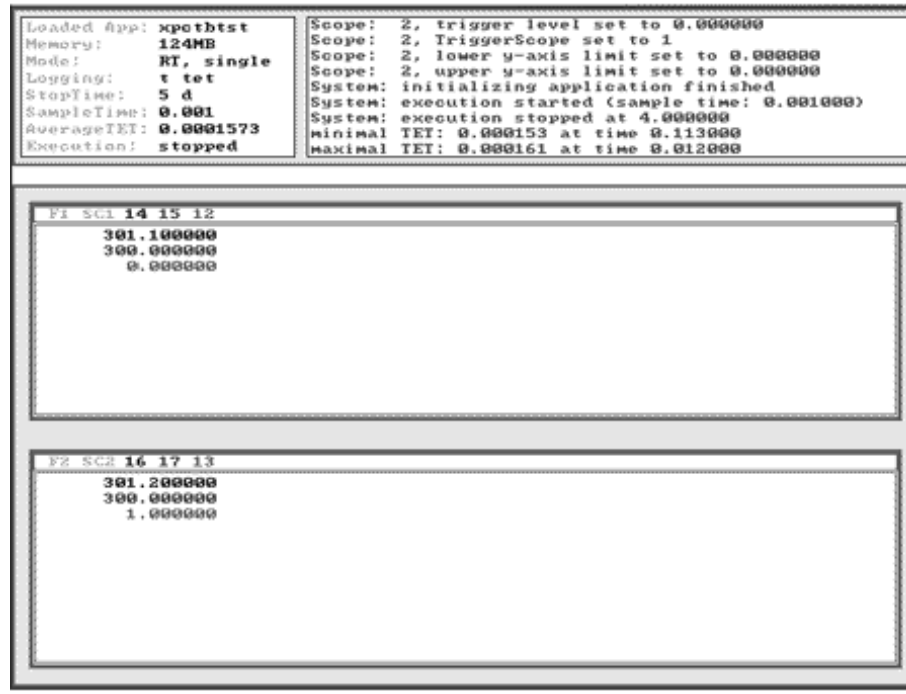
- A numerical value of `1` indicates that the loop-back self-test was successful for that particular I/O option. If all numerical values are `1`, the self-test is running successfully and your xPC TargetBox is fully functional. Unplug the power cord and proceed to Chapter 3, “Regular Use.”
- If one of the numerical values is `0`, the loop-back test detected a failure when exercising a particular I/O option. Unplug the power cord and

proceed to the following topic, “Troubleshooting the Loop-Back Self-Test” on page 2-12.

Troubleshooting the Loop-Back 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 screen below shows that I/O option 301 analog (`301.10000`) failed (`0.000000`) 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.

Board Name	IO Option
Diamond-MM-32-AT	IO 301
Diamond Ruby-MM-1612	IO 302
Diamond Ruby-MM-416	IO 303
Diamond Onyx-MM	IO 304
Diamond Quartz-MM-10	IO 305
Real Time Devices DM6814	IO 306
Softing CAN-AC2-104	IO 308

5 Turn the xPC TargetBox over and find the configuration label on the bottom. Look for a listing of I/O options for this box. 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" on page 2-15 — 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" on page 2-16 — Identify all I/O options and associated I/O connectors
- "Planning I/O Hardware Connections" on page 2-22— Read hardware manuals and draw a diagram of the connections
- "Connecting Hardware to Screw Terminal Boards" on page 2-23 — 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.

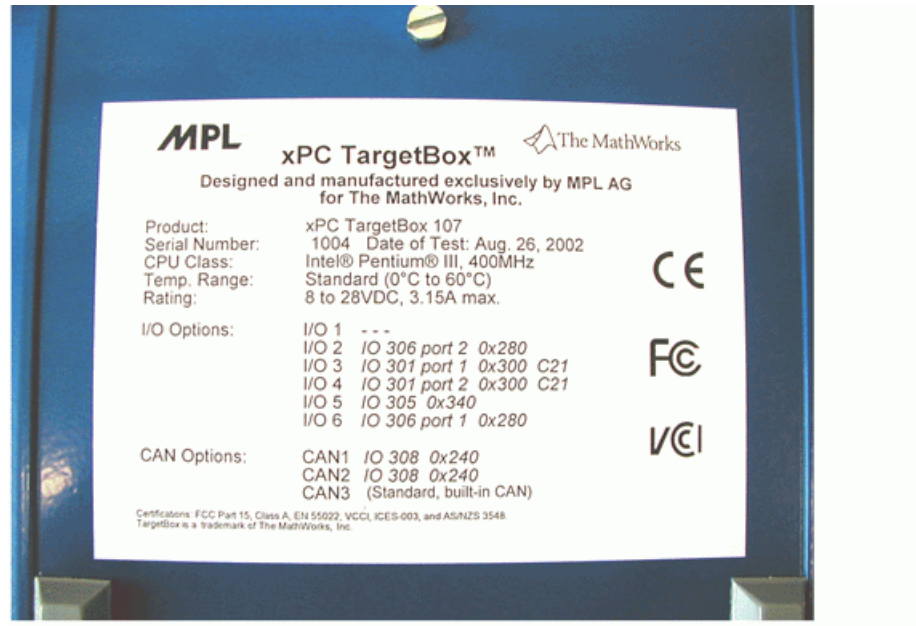
For information about the pin layout for a specific I/O option, see Chapter 5, "xPC TargetBox I/O Options."

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 below.

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. For a description of the I/O options, see "xPC TargetBox I/O Options" on page 1-14.

Jumper Codes for I/O Options

I/O Option	Number of Connectors	Jumper Codes
IO 301 Diamond-MM-32-AT	2	**Analog input (A/D) jumper codes** (first number after the C) • C1# — 32 single-ended • C2# — 16 differential • C3# — 16 single-ended/8 differential
		Analog output (D/A) jumper codes (second number after the C indicates the output voltage range for the 4 D/A channels) • C#1 — ±10 V • C#2 — ±5 V • C#3 — 0-10 V • C#4 — 0-5 V
		The default configuration is C21, selecting 16 differential input channels and ±10 V for the output range on the 4 output channels.

Jumper Codes for I/O Options (Continued)

I/O Option	Number of Connectors	Jumper Codes (Continued)
IO 302 Ruby-MM-1612	1	**Analog output (D/A) jumper codes** (first number after the C indicates the output voltage range for the first 8 D/A channels) • C1# — ±10 V • C2# — ±5 V • C3# — ±2.5 V • C4# — 0-10 V • C5# — 0-5 V • C6# — 0-2.5 V The third character (second number) in the code indicates the output voltage range for the second 8 channels. The default configuration is C11, selecting ±10 V for the output range on the 16 output channels. The following lists the allowed output voltage range combinations for the two channel banks. **B1** is the first bank of channels, **B2** is the second bank of channels.

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

Jumper Codes for I/O Options (Continued)

I/O Option	Number of Connectors	Jumper Codes (Continued)		
		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 for I/O Options (Continued)

I/O Option	Number of Connectors	Jumper Codes (Continued)
IO 303 Ruby-MM-416	1	**Analog output (D/A) jumper codes** (first number after the C indicates the output voltage range for the first D/A channel) • C1### — ±10 V • C2### — ±5 V • C3### — 0-10 V The third, fourth, and fifth characters in the code indicate the output voltage range for the second, third, and fourth D/A channels. The default configuration is C1111, selecting ±10 V for the output range on the four output channels.
IO 304 Onyx-MM	2	None
IO 305 Quartz-MM-10	1	None
IO 306 DM6814	2	None
IO 308 CAN	Special connectors	None

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 2-22.

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 TargetBox 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

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

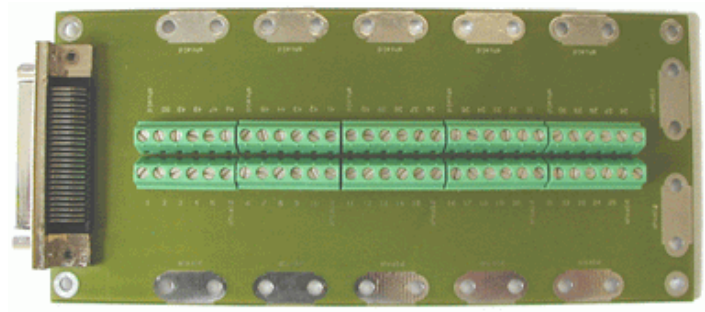
Connecting Hardware to Screw Terminal Boards

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.



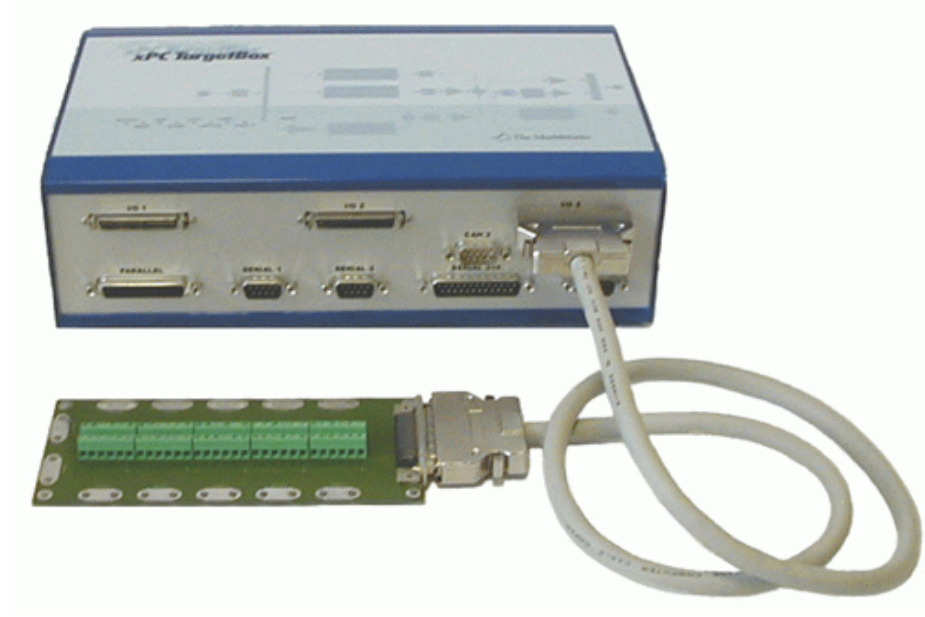
2. You can find the I/O cables in the lower compartment of the shipping case.

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 loop-back testing of an I/O option before connecting it to your real hardware. See Chapter 5, “xPC TargetBox I/O Options.”

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

3

Regular Use

Regular use of xPC Target with an xPC TargetBox uses a connection to a separate host PC and a floppy disk drive to boot the xPC TargetBox. This chapter includes the following sections:

xPC Target Software Installation (p. 3-2)	Install the xPC Target software and activate the additional boot modes. Optionally, install the xPC Target Embedded Option to use the API and COM API interfaces.
xPC TargetBox Hardware Installation (p. 3-6)	Remove the test dongles and connect a floppy disk drive.
Host PC to xPC TargetBox Communication (p. 3-12)	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. 3-18)	Boot the xPC Target kernel on an xPC TargetBox and establish a connection with the host PC. Test connections and communication between the host PC and an xPC TargetBox 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" on page 3-2 — Install 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 Boot Modes" on page 3-3 — With an xPC TargetBox, but without xPC Target Embedded Option, activate DOSLoader and StandAlone modes.
- "xPC Target Embedded Option Software" on page 3-4 — 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:

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

For information on installing software on the host computer, see Chapter 2, "Installation and Configuration," in the xPC Target Getting Started documentation.

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

Activating the Additional 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, StandAlone).

Note Even if you do not have an xPC TargetBox, you can activate the additional boot modes and use the setup dialog box 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.

---

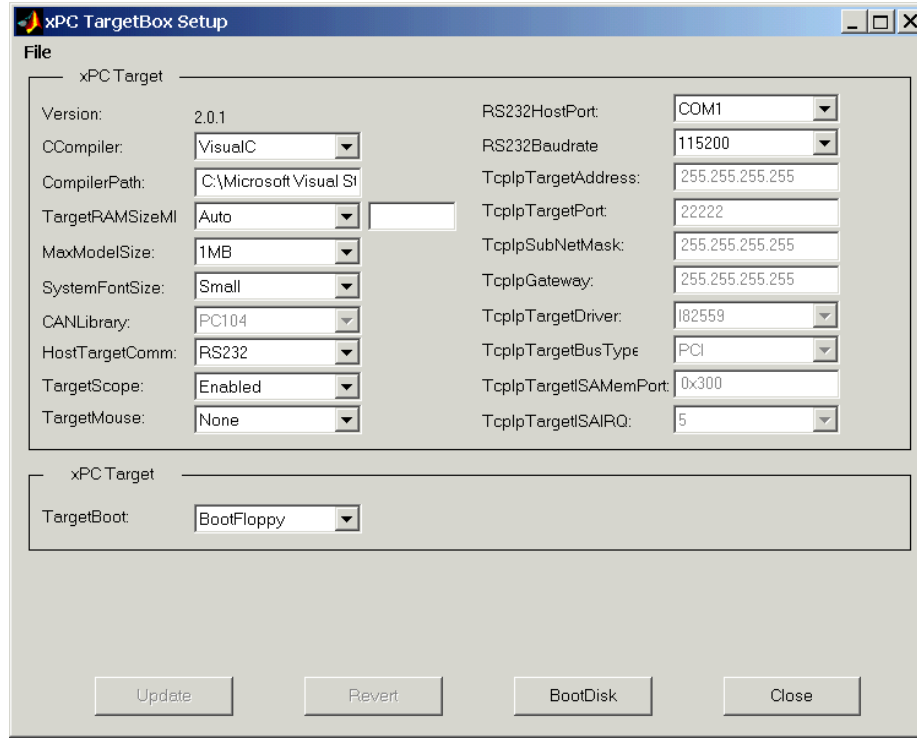
1. On the host computer, start MATLAB.
2. In the MATLAB Command Window, type

   ```
   xpctargetboxinit
   ```

   The additional boot modes are now active in the **xPC TargetBox Setup** dialog box.
3. Type

   ```
   xpcsetup xpctargetbox
   ```

   The **xPC TargetBox Setup** dialog box specific for an xPC TargetBox opens.



For information on entering and selecting parameters for the setup dialog box, see “Host PC to xPC TargetBox Communication” on page 3-12.

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 features with an xPC TargetBox:

- **API** — Create GUI applications that interface with a target application.
- **COM API** — Create GUI applications that communicate with a target application using a COM interface.
- **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 Chapter 2, “Installation and Configuration,” in the xPC Target Getting Started documentation.

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" on page 3-6 — Remove and save the dongles to troubleshoot I/O boards later
- "Connecting the External Floppy Disk Drive" on page 3-7 — Connect the external floppy drive and copy files from the host PC to an xPC TargetBox
- "Connecting Additional Peripherals" on page 3-9—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, and with DOSLoader and StandAlone modes, to copy the xPC Target kernel and target application to the flash disk. With DOSLoader and StandAlone modes, 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.

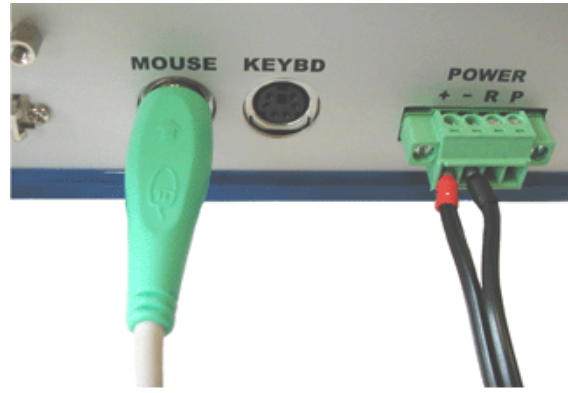
The next step is to connect the xPC TargetBox to a host computer using either a serial or network connection. See “Host PC to xPC TargetBox Communication” on page 3-12.

Connecting Additional Peripherals

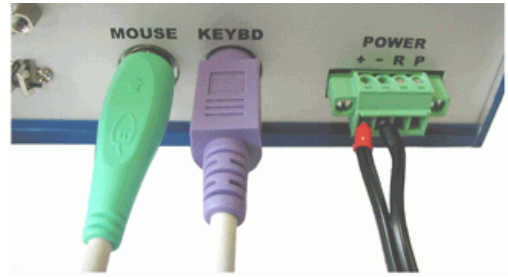
When using the boot modes BootFloppy and DOSLoader, the xPC TargetBox is connected to a host PC. From this host PC you can control and interact with a target application running on an xPC TargetBox.

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 the xPC Target User's Guide documentation.

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" on page 3-12 — Connect a null modem cable
- "Environment Properties for Serial Communication" on page 3-12 — Enter the host PC COM port and baud rate
- "Hardware for Network Communication" on page 3-14 — Connect to a LAN or connect a crossover cable
- "Environment Properties for Network Communication" on page 3-15 — 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 COM1 or COM2 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 and target computers. See "xPC Target Software Installation" on page 3-2.

1 In the MATLAB Command Window, type

```
xpcsetup xpctargetbox
```

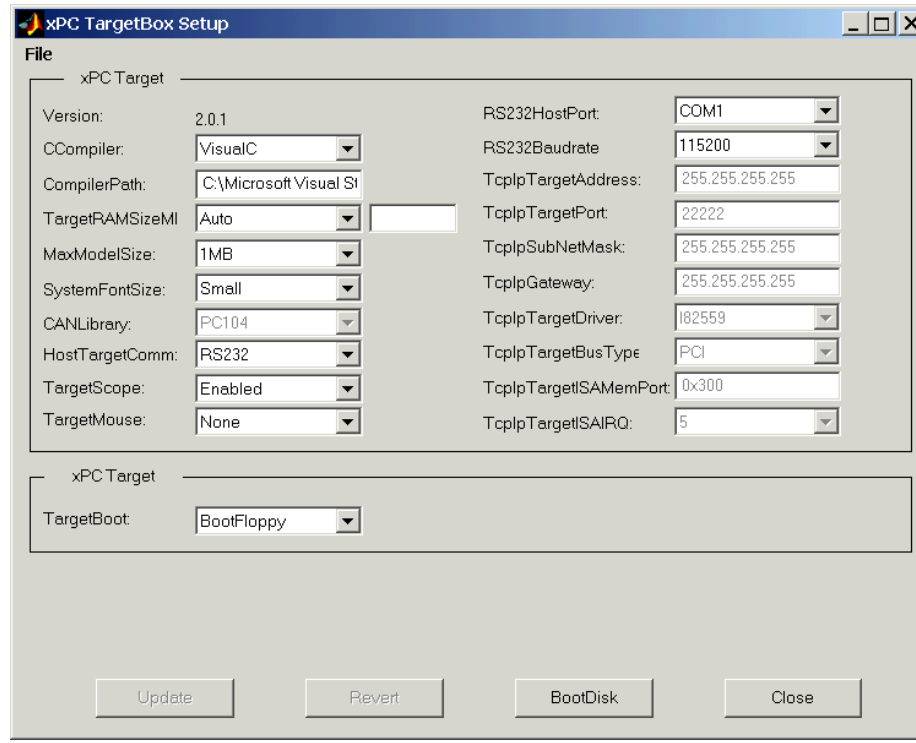
The **xPC TargetBox Setup** dialog box opens.

2 From the **CCompiler** list, select either `VisualC` or `Watcom`.

3 In the **CompilerPath** box, enter the root path where you installed your C/C++ compiler.

4 From the **HostTargetComm** list, select `RS232`.

5 From the **RS232HostPort** list, select either `COM1` or `COM2` for the connection on the host PC. xPC Target automatically determines the COM port you use on an xPC TargetBox.



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

xPC Target updates the environment with the new properties.

You do not need to exit and restart MATLAB after making changes to the xPC Target environment, even if you change the communication between the host and target from RS-232 to TCP/IP. However, 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 Chapter 3, "Software Environment," in the xPC Target User's Guide documentation.

Your next task is to create a target boot disk. See "Creating a Target Boot Disk" on page 3-18.

Hardware for Network Communication

You must install the following hardware before you use the xPC Target software 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 and target computers. See "xPC Target Software Installation" on page 3-2.

1. In the MATLAB Command Window, type

   ```
   xpcsetup xpctargetbox
   ```

   The **xPC TargetBox Setup** window opens.

2. From the **CCompiler** list, select either `VisualC` or `Watcom`.

3. In the **CompilerPath** box, enter the path to where you installed your C/C++ compiler.

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

   The TCP/IP text boxes become active.

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

   - **TcpIpTargetAddress** — This is the IP address for your xPC TargetBox. An example of an IP address is `192.168.0.1`.
   - **TcpIpSubNetMask** — This is the Subnet Mask address of your LAN. An example of a Subnet Mask address is `255.255.255.0`.

   You enter the following properties depending on your specific circumstances:

   - **TcpIpTargetPort** — This property is set by default to `22222`. This value should not cause any problems, because this number is higher than the reserved area (`telnet`, `ftp`, ...) and it is only of relevance on an xPC TargetBox. If necessary, you can change this property value to any value higher than `20000` and less than `65536`.

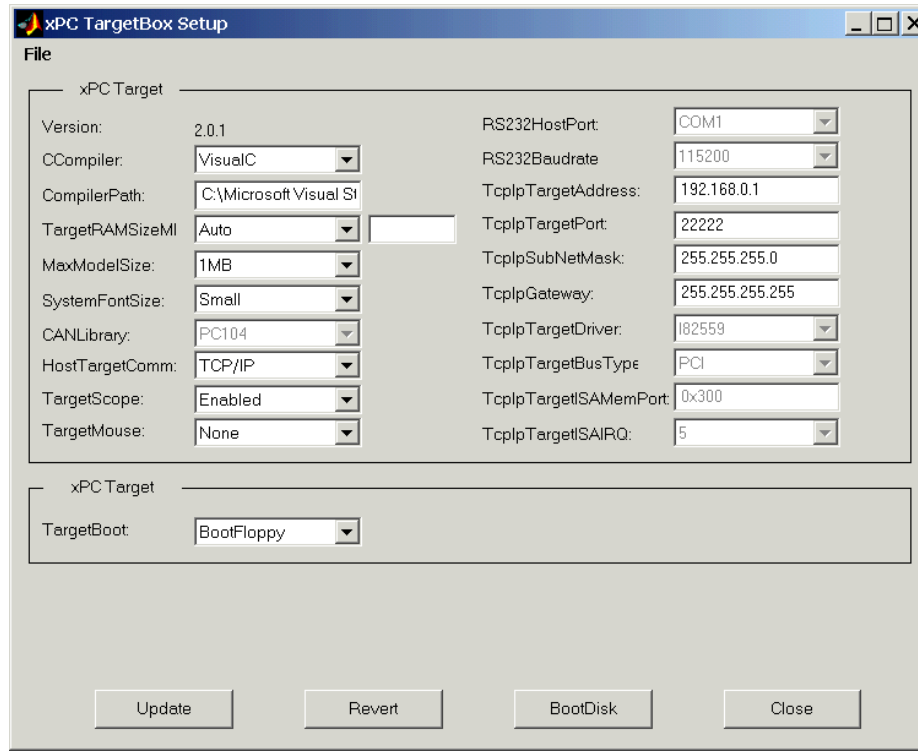
- **TcpIpGateway** — This property is set by default to `255.255.255.255`. This means that you do not use a gateway to connect to your xPC TargetBox.

  If you communicate with an 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.

The following properties are specific for the Ethernet controller on your xPC TargetBox:

- **TcpIpTargetDriver** — Because the xPC TargetBox includes an Intel 82559ER Ethernet controller, `I82559` is selected for you and cannot be changed.
- **TcpIpTargetBusType** — `PCI` is selected.
- **TcpIpISAMemPort** and **TcpIpISAIRQ** — Because the xPC TargetBox uses a PCI Ethernet controller, you do not need to enter values for these properties.



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

xPC Target updates the environment with the new properties.

You do not need to exit and restart MATLAB after making changes to the xPC Target environment, even if you change the communication between the host and target from RS-232 to TCP/IP. However, 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 Chapter 3, “Software Environment,” in the xPC Target User’s Guide documentation.

Your next task is to create a target boot disk. See “Creating a Target Boot Disk” on page 3-18.

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 target boot disk also includes the target application. This section includes the following topics:

- "Creating a Target Boot Disk" on page 3-18 — Use the `xpcsetup` command to open a dialog box and create a boot disk
- "Booting an xPC TargetBox" on page 3-20 — Check xPC TargetBox connections to peripherals and power supply
- "Testing the Installation" on page 3-23 — Run the `xpctest` command
- "Test 1, Ping Target System Standard Ping" on page 3-24 — Test network communication with a standard ping command
- "Test 2, Ping Target System xPC Target Ping" on page 3-26 — Test either network or serial communication with the host PC
- "Test 3, Reboot Target Using Direct Call" on page 3-26 — Test rebooting an xPC TargetBox from the host PC
- "Test 4, Build and Download Application" on page 3-27 — 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:

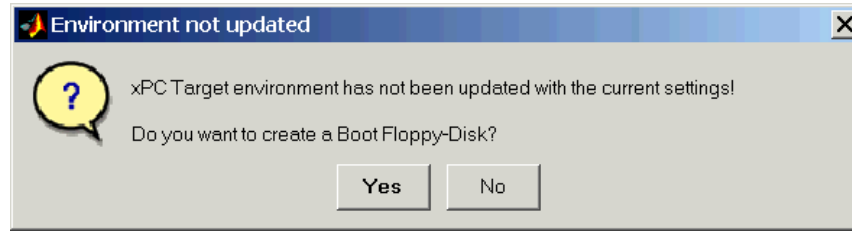
1 In the MATLAB Command Window, type

```
xpcsetup xpctargetbox
```

The **xPC TargetBox Setup** window opens.

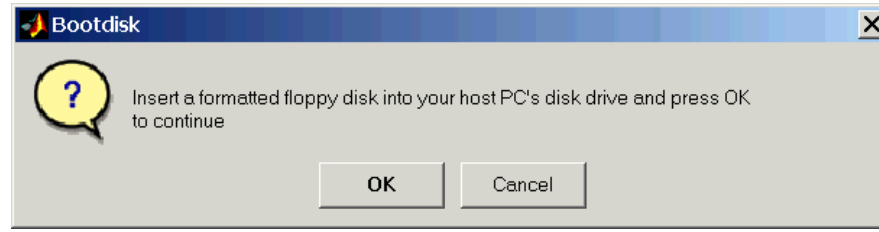
2 Click the **BootDisk** button.

If you did not update the current settings, the following message box opens.



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

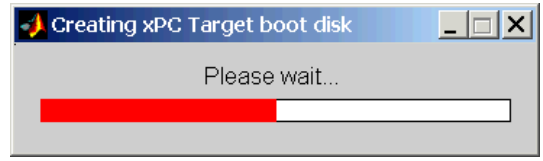
After you update the current properties and click the **BootDisk** button, the following message box opens.



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

Note: All data on the disk is erased.

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



4 Close the **xPC TargetBox Setup** window.

5 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 3-20.

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 TargetBox Setup** dialog box, you are ready to power up the xPC TargetBox again. This is a good time to check the xPC TargetBox connections.

1 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.

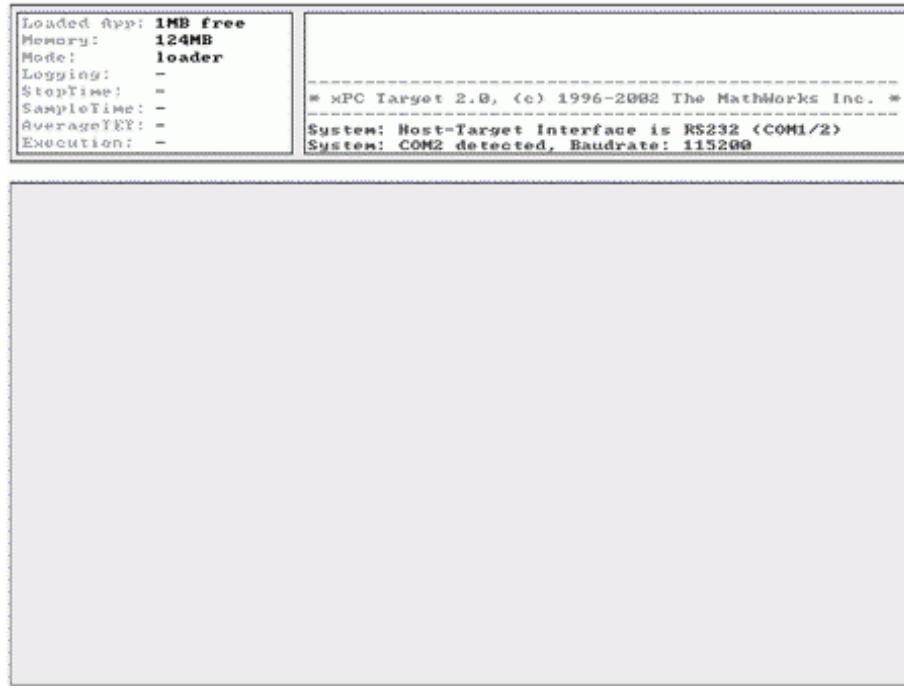
2 Insert an xPC Target boot disk into the external drive.

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.

Your next step is to test the installation. See “Testing the Installation” on page 3-23.

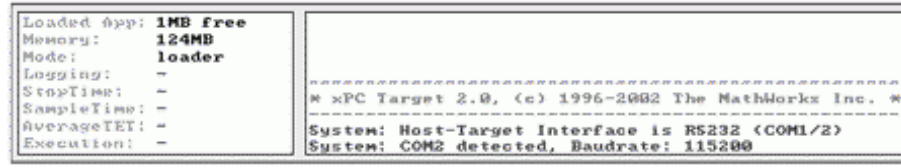
Testing the Installation

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.

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

1. Insert your target boot disk into the external disk drive.
2. Press the **Reset** button on the xPC TargetBox.

   After loading the BIOS, xPC Target boots the kernel and displays the following screen on the xPC TargetBox monitor.



3. 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.

4. In the MATLAB Command Window, type

   ```
   xpctest
   ```

   MATLAB runs the test script and displays messages indicating the success or failure of a test. If you use RS-232 communication, the first test is skipped.

   ```
   ### xPC Target Test Suite 2.0.1
   ```

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

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

- "Test 1, Ping Target System Standard Ping" on page 3-24
- "Test 2, Ping Target System xPC Target Ping" on page 3-26
- "Test 3, Reboot Target Using Direct Call" on page 3-26
- "Test 4, Build and Download Application" on page 3-27

If all of 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 2-15.
- Build and download a target application to the xPC TargetBox. See Chapter 3, "Basic Tutorial," in the xPC Target Getting Started documentation.

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 the **xPC TargetBox Setup** window.

To solve this problem, open the **xPC TargetBox Setup** dialog box. In the MATLAB Command Window on the host PC, type

```
xpcsetup xpctargetbox
```

Check that **TcpIpTargetAddress**, **TcpIpSubNetMask**, and **TcpIpGateway** have the correct values. Click the **Update** button 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 the **xPC TargetBox Setup** window.

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

```
xpcsetup xpctargetbox
```

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

If you still cannot solve your problem, see “If You Still Need More Help” on page 3-27.

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=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 3-18.

If you still cannot solve your problem, see “If You Still Need More Help” on page 3-27.

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

```
xpctest noreboot
```

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

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

2 Observe the messages in the MATLAB Command Window during the build process.

If you still cannot solve your problem, see "If You Still Need More Help" on page 3-27.

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 the xPC Target setup dialog box. Type

```
xpcsetup 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 3-27.

If You Still Need More Help

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

Internet	`http://www.mathworks.com/support/`
E-mail	`mailto:support@mathworks.com`
Telephone	508-647-7000 Ask for Technical Support.

4

Advanced Use

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).

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" on page 4-2 — Drivers from the xPC Target library are conveniently grouped into an xPC TargetBox library
- "Using the Panel LEDs" on page 4-3 — Control the two LEDs on the xPC TargetBox using driver blocks
- "Using the Watchdog Timer" on page 4-4 — Perform a system reset when a programmable timeout occurs

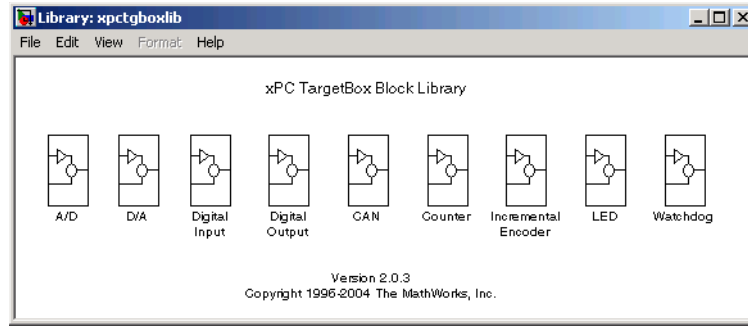
Drivers in the xPC TargetBox Library

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 documentation 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 reset 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` seconds.

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 system when the watchdog timer expires and asserts IRQ15.

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 flash disk, remove the floppy disk drive, and then boot the xPC Target kernel from the flash disk. The target application is still downloaded from the host PC.

Use this mode for applications where an xPC TargetBox is not easily accessible. This section includes the following topics:

- "Updating Environment Properties and Creating a Boot Disk" on page 4-6 — Select `DOSLoader` mode in the **Setup** dialog box
- "Copying the Kernel to Flash Memory" on page 4-7 — Copy the xPC Target kernel to the flash disk on an xPC TargetBox and then start the kernel running
- "Creating a Target Application" on page 4-9 — Create, download, and run a target application from a host PC

Note `DOSLoader` mode is included with xPC TargetBox when using the xPC TargetBox setup dialog. To use `DOSLoader` mode with another target PC, you need to use the xPC Target Embedded Option. See the xPC Target documentation.

Updating Environment Properties and Creating a Boot Disk

xPC Target uses the environment properties to determine what files to create for the various target boot modes.

This procedure assumes you have serial or network communication working correctly between your host computer and an xPC TargetBox. It would be helpful to successfully create a target application with the **TargetBoot** option in the **xPC TargetBox Setup** dialog set to `BootFloppy` before trying to create a kernel that boots from DOS.

1 On the host computer, start MATLAB.

2 In the MATLAB Command Window, type

```
xpcsetup xpctargetbox
```

The **xPC TargetBox Setup** window opens.

3 From the **TargetBoot** list, choose `DOSLoader`. Click **Update**.

xPC Target updates the environment properties and grays the **Update** button.

4 Click **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.
```

5 Insert a floppy disk, and then click **OK**.

The files `checksum.dat`, `xpcsgo1.rtb`, `xpcboot.com`, and `autoexec.bat` are copied to the disk.

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 4-6:

1 If there is a floppy disk in the external disk drive, remove it. 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)?
```

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

3 Type `Y`.

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

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, 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
```

Alternatively, you can copy the file `c:\autoexec.wrk` to `c:\autoexec.bat`. For example,

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

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

a `c:\autoexec.bat`

b `c:\work\autoexec.bat`

c `c:\work\xpcboot.com`

d `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 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. This section includes the following topics:

- "Updating Environment Properties" on page 4-10 — Select StandAlone mode in the **Setup** dialog box
- "Creating a Kernel/Target Application" on page 4-11 — On the host PC, create a stand-alone application
- "Copying the Kernel/Target Application to Flash Memory" on page 4-12 — Copy the combined xPC Target kernel and target application to the flash disk on an xPC TargetBox

---

Note StandAlone mode is included with xPC TargetBox when using the **xPC TargetBox Setup** dialog. To use StandAlone mode with another target PC, you need to use the xPC Target Embedded Option. See the xPC Target documentation.

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 serial or network communication working correctly between your host computer and an xPC TargetBox. It would be helpful to successfully create a target application with the **TargetBoot** option in the **xPC TargetBox Setup** dialog set to `BootFloppy` before trying to create a stand-alone application.

1 On the host computer, start MATLAB.

2 In the MATLAB Command Window, type

```
xpcsetup xpctargetbox
```

The **xPC TargetBox Setup** window opens.

3 From the **TargetBoot** list, choose `StandAlone`.

4 Click **Update**.

xPC Target updates the environment properties and 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 onto a formatted floppy 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
```

3 Copy the preceding files to a formatted floppy disk.

Copying the Kernel/Target Application to Flash Memory

You build a target application on a host PC using Real-Time Workshop, xPC Target, and a C/C++ compiler. One method for transferring the files from the host PC to an xPC TargetBox is to use an external floppy disk drive.

After you build a stand-alone application on a host PC, you can copy files from a floppy disk to the flash disk. See "Creating a Kernel/Target Application" on page 4-11.

1. If there is a floppy disk in the external disk drive, remove it. 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.

4. Insert the boot disk with the stand-alone application and utility files into the external floppy disk drive.

5. Type

   ```
   copy a:\xpc_osc3.rtb c:\work
   copy a:\xpcboot.com c:\work
   copy a:\autoexec.bat c:\work
   ```

6. If you want your stand-alone application to run when you press the **Reset** button on your target PC, 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
   ```

   Alternatively, you can copy the file `c:\autoexec.wrk` to `c:\autoexec.bat`. For example,

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

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 and stand-alone application. 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, you manually added a `config.sys` file and you should know the location of this file.

The sequence of calls during the boot process is

a `c:\autoexec.bat`

b `c:\work\autoexec.bat`

c `c:\work\xpcboot.com`

d `c:\work\<application>.rtb`

8 On the xPC TargetBox keyboard, press the spacebar.

A command line opens on the xPC TargetBox screen.

For a complete list of target PC commands, see the xPC Target User's Guide documentation.

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" on page 4-15 — Description of the directories and files on the flash disk
- "Booting xPC TargetBox with FTP Server" on page 4-17 — Edit the file `autoexec.bat` to boot the xPC TargetBox so that the FTP server is running
- "Using FTP Commands" on page 4-18 — List of common FTP commands you enter from the host PC
- "Copying Files with DOS and FTP" on page 4-19— Use FTP to copy stand-alone applications from a host PC to an xPC TargetBox
- "Copying Files with MATLAB and FTP" on page 4-21 — Use an xPC Target function from MATLAB to establish an FTP connection and copy stand-alone applications from a host PC to an xPC TargetBox
- "Booting xPC TargetBox to DOS" on page 4-22 — Edit the file `autoexec.bat` to boot the xPC TargetBox into DOS

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. See "Contacting The MathWorks for Technical Support" on page xvi.

Directory or Filename	Description
`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>`
`  doststst.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

Directory or Filename	Description

`ftpbin.exe`
`telpass.exe`
`config.tel`
`password.tel`
	Files to start the FTP server (`ftpbin.exe`), set IP address (`config.tel`), and change password (`telpass.exe`)

`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 4-10.

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, 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
```

Alternatively, you can copy the file `c:\autoexec.wrk` to `c:\autoexec.bat`. For example:

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

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.

Copying Files with MATLAB and FTP

As an alternative to transferring files to an xPC TargetBox using a DOS shell and FTP, you can use MATLAB.

After you build a stand-alone application, you can transfer the application and utility files to an xPC TargetBox:

1 Boot the xPC TargetBox with the FTP server. See "Booting xPC TargetBox with FTP Server" on page 4-17.

The system boots the FTP server.

2 In the MATLAB Command Window, type

```
xpcsetup xpctargetbox
```

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

3 Enter information for a network connection. See "Hardware for Network Communication" on page 3-14.

4 Transfer a stand-alone application and utility files to an xPC TargetBox. For example, if your application is `xpcosc.rtb`, type

```
xpctgboxdl('xpcosc')
```

An FTP connection is established between the host PC and the xPC TargetBox. The application (`xpcosc.rtb`) and the utility files (`autoexec.bat, xpcboot.com`) are downloaded from `<MATLAB current directory>\xpcosc_xpc_emb` to the directory `c:\work`.

If the download was successful, MATLAB displays a confirming message. Otherwise, MATLAB displays an error message.

5 Run the target application. From the xPC TargetBox DOS prompt, 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.

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.

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.

5

xPC TargetBox I/O Options

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

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" on page 5-2 — Pin numbers on a screw terminal board match the pin numbers on an xPC TargetBox connector
- "Making Your Own Terminal Boards" on page 5-2 — Use custom hardware connection to add application-specific signal conditioning circuits

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.

Loop-Back Testing of I/O Options

Use a loop-back 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:

- "Loop-Back Testing Process" on page 5-3 — 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 Loop-Back Test" on page 5-4 — Observe a scope on the xPC TargetBox monitor that displays the results from a loop-back test
- "Uses for Loop-Back Testing" on page 5-4 — Test your xPC TargetBox when you first unpack the box, during use, and to learn about an I/O option
- "Creating a Simulink Model for Loop-Back Testing" on page 5-5 — 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" on page 5-6 — Create and run a target application to test I/O options

Loop-Back Testing Process

Loop-back 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 Loop-Back Test

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

Uses for Loop-Back Testing

You can use loop-back testing of an I/O option to accomplish various testing tasks at different stages when using an xPC TargetBox. Some uses of loop-back 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 loop-back 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 Loop-Back Testing

During the final system test at the manufacturer, a Simulink model was created that does a loop-back 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 2-16 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({'IO301','0x300','C21'},{'IO305','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 loop-back 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 ± 10 volts, type

```
xpctgboxtest({'IO301','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 loop-back test was successful and the problem with your application might lie 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. This section includes the following topics:

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

Wiring for IO 301 Test Dongles

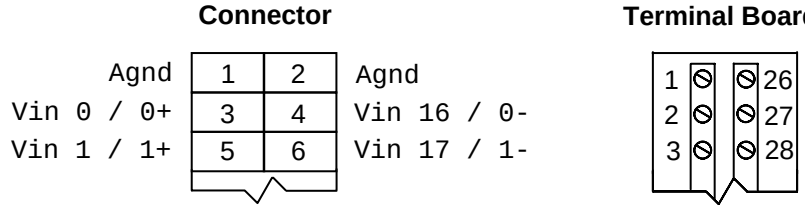
xPC TargetBox communicates with this board using two connectors. Some of the pins in the test dongles are connected together to allow for loop-back testing of this board:

- 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.

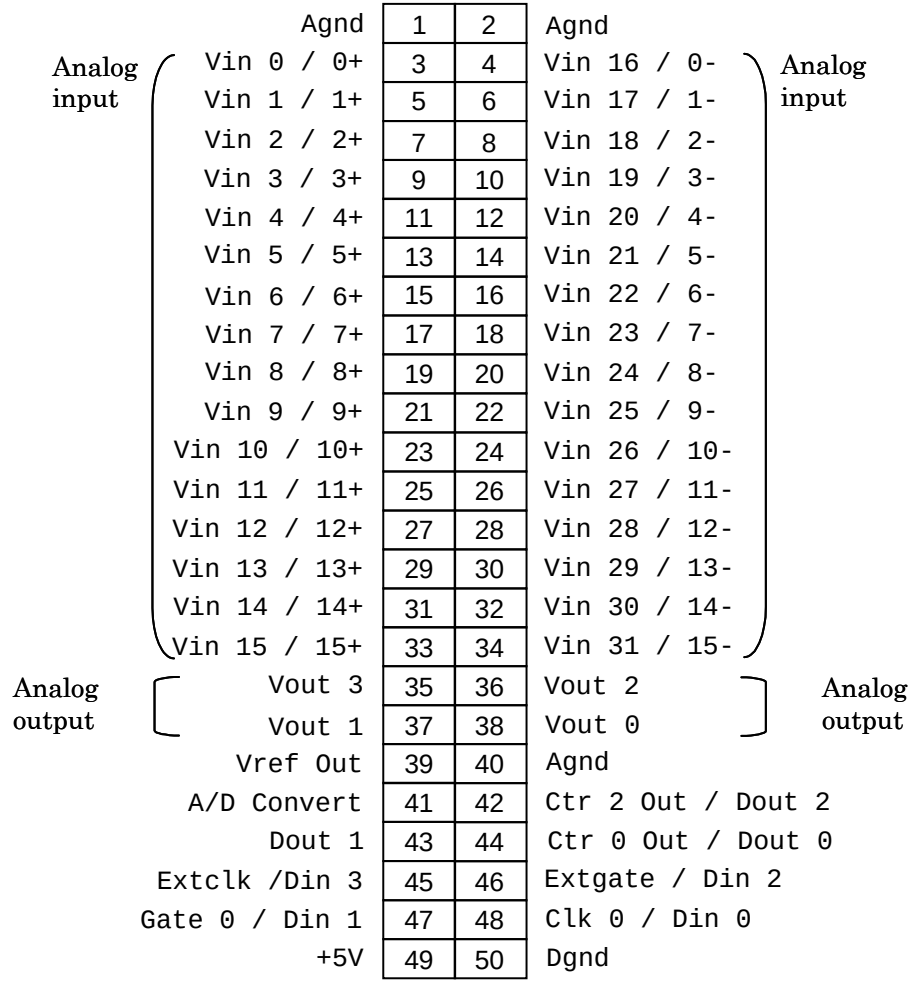
However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the connector figures.



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/DMM32v2.61.pdf`.

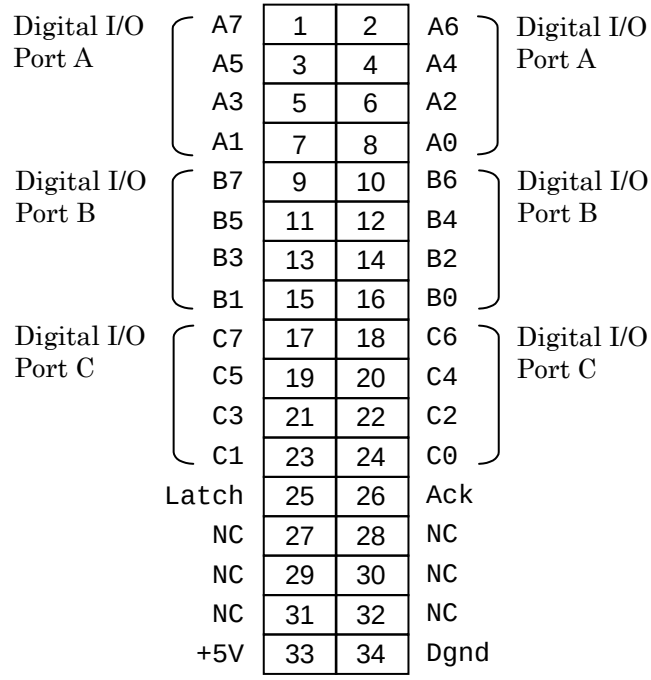
The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.



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/DMM32v2.61.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.



Note This connector does not use pins 35 to 50.

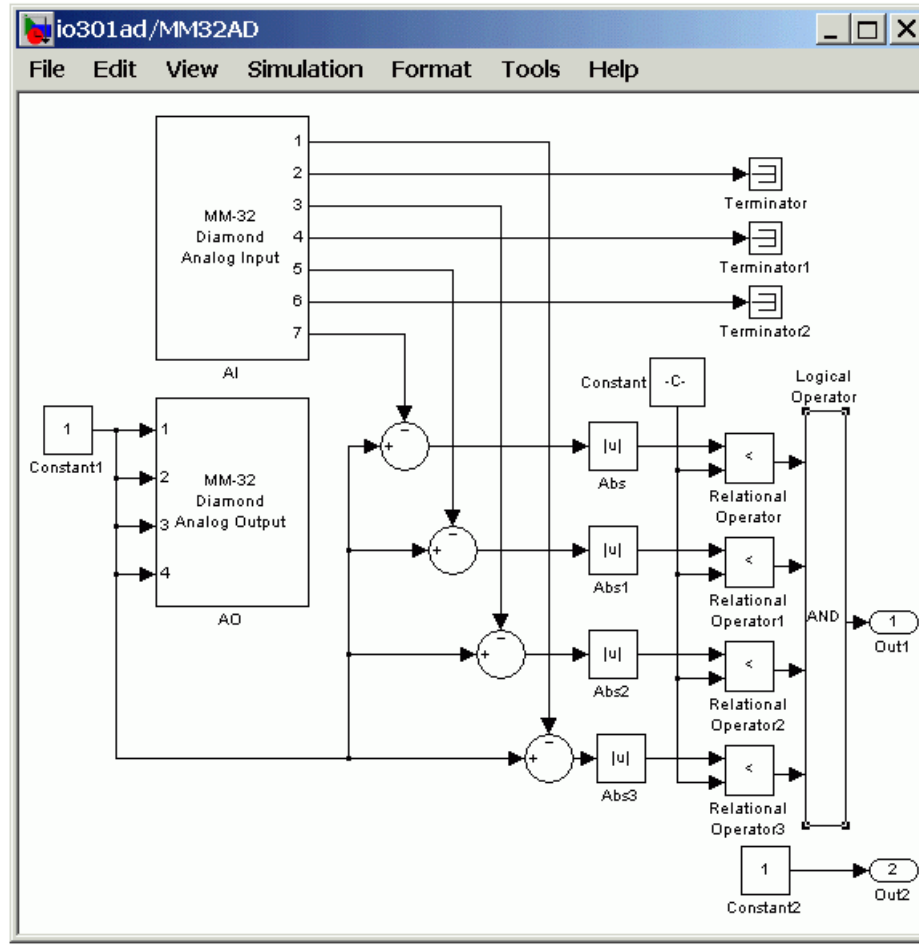
Testing Model IO 301

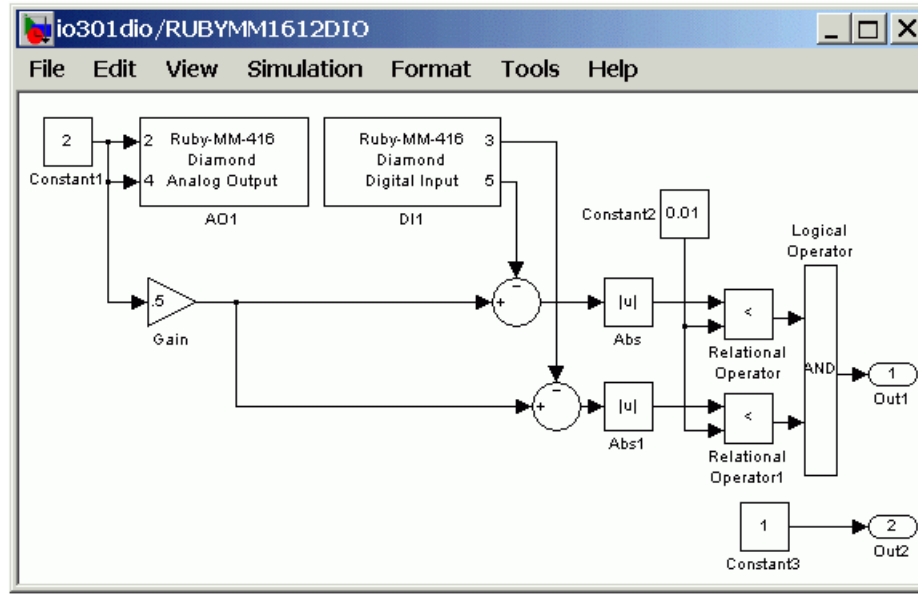
This model tests the analog channels. A constant value is written to an A/D driver block and then read from the D/A driver block. The values are compared with an error defined in a second constant block:

1 In the MATLAB Command Window, type

```
io301
```

The Simulink model for testing the IO 301 option opens.




io301dio/RUBYMM1612DIO
File
Edit
View
Simulation
Format
Tools
Help
2
Constant1
2 Ruby-MM-416
Diamond
4 Analog Output
AO1
Ruby-MM-416
Diamond
Digital Input
3
5
DI1
Constant2
0.01
Logical
Operator
.5
Gain
|u|
Abs
<
Relational
Operator
AND
1
Out1
|u|
Abs1
<
Relational
Operator1
1
Constant3
2
Out2

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:

- "Diamond-MM-32-AT Analog Input (A/D, IO 301)"
- "Diamond-MM-32-AT Analog Output (D/A, IO 301)"
- "Diamond-MM-32-AT Digital Input (IO 301)"
- "Diamond-MM-32-AT Digital Output (IO 301)"

Board Characteristics

Board name	Diamond-MM-32-AT
Manufacturer	Diamond Systems Corporation
Bus type	ISA (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 lie 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 Analog Output (D/A, IO 301)

Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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.

Input Range (V)	Range Code
-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 channel is 0 to + 10 volts and the second channel is 0 to +5 volts, enter

```
[10,5]
```

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 or 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:

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

Wiring for IO 302 Test Dongle

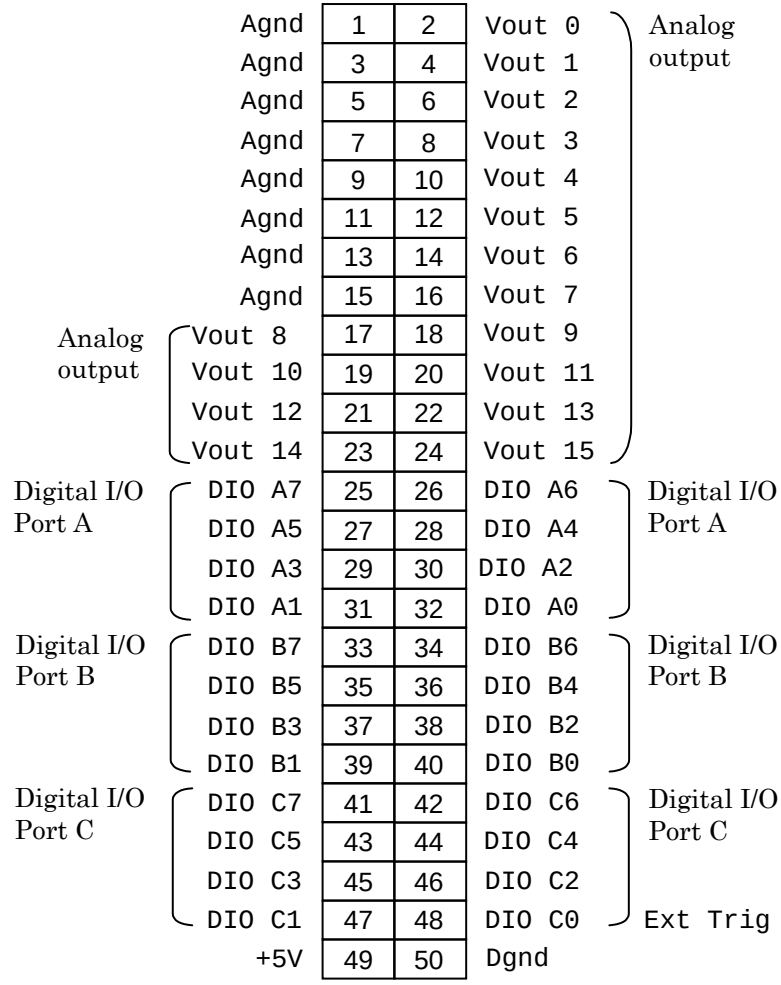
xPC TargetBox communicates with this board using one connector. Some of the pins in the test dongle are connected together to allow for loop-back testing of this board.

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.



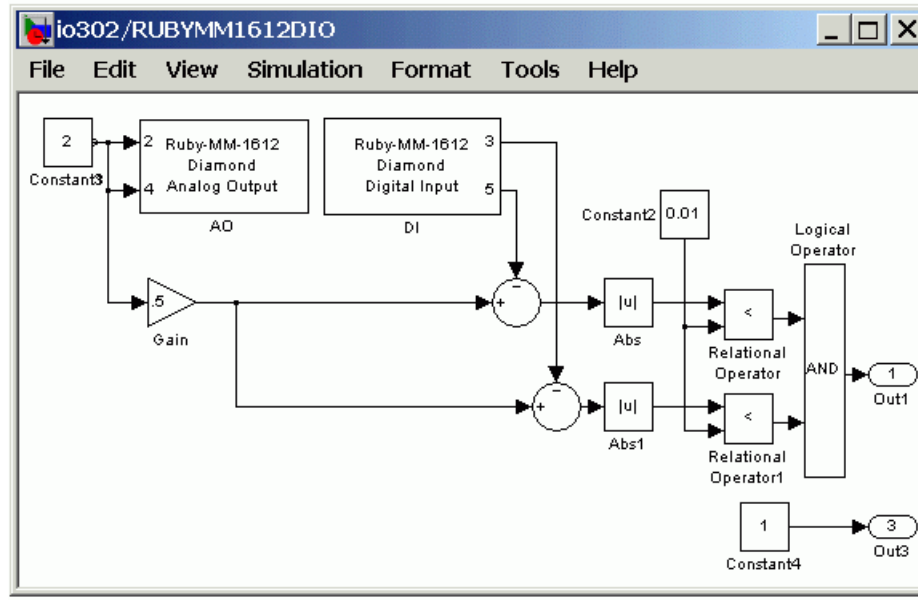
Testing Model IO 302

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 Outport 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)"
- "Ruby-MM-1612 Digital Input (IO 302)"
- "Ruby-MM-1612 Digital Output (IO 302)"

Board Characteristics

Board name	Ruby-MM-1612
Manufacturer	Diamond Systems Corporation
Bus type	ISA (PC/104)
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 V = 1.1 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 of 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:

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

Wiring for IO 303 Test Dongle

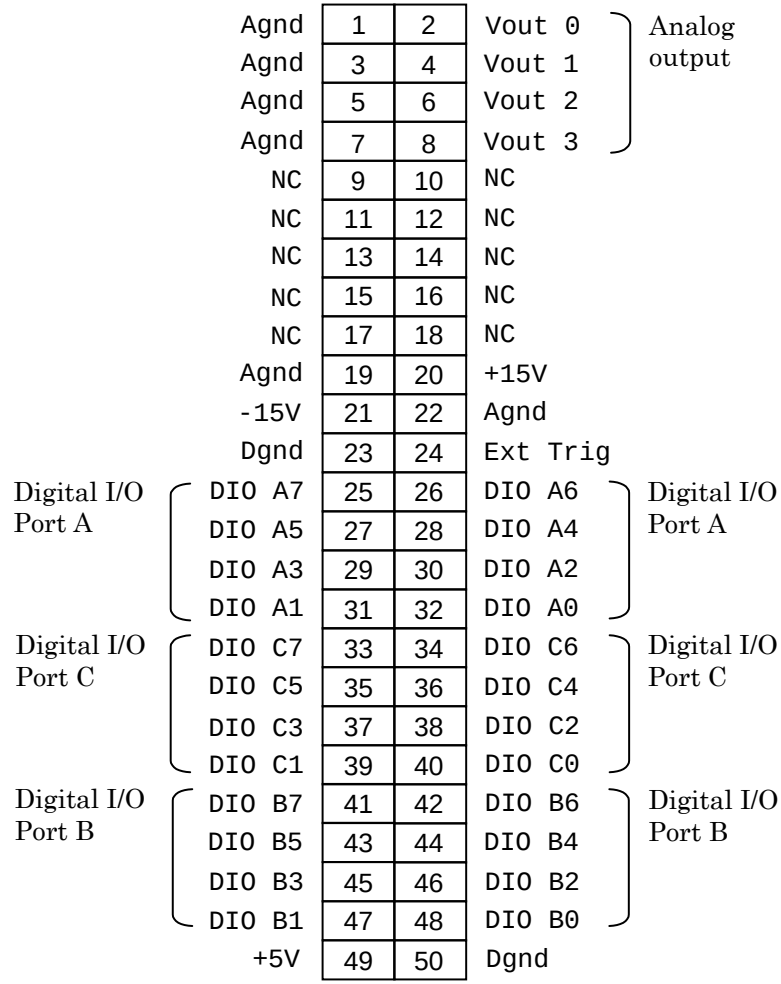
xPC TargetBox communicates with this board using one connector. Some of the pins in the test dongle are connected together to allow for loop-back testing of this board.

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.



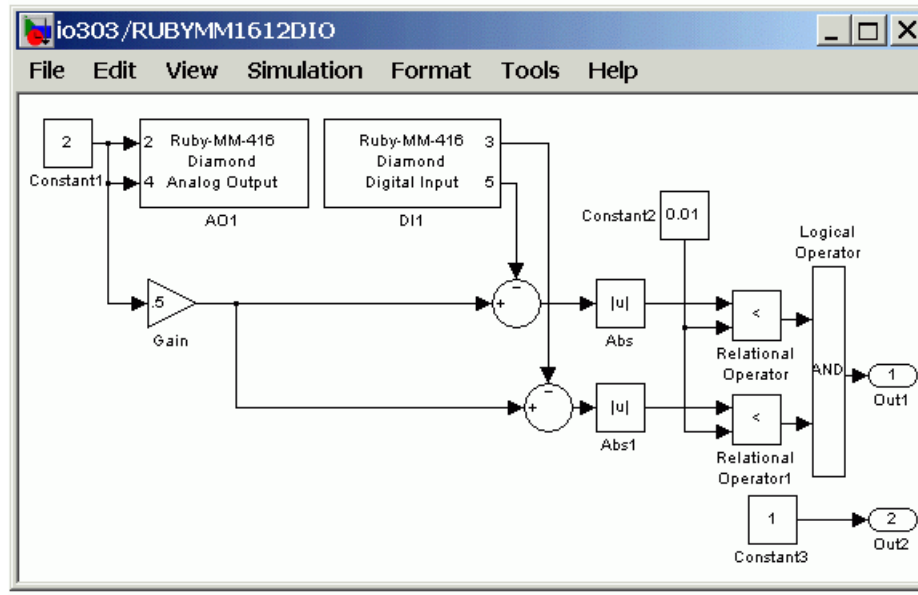
Testing Model IO 303

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 Outport block:

1 In the MATLAB Command Window, type

```
io303
```

The Simulink model for testing the IO 303 option opens.



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 5-32
- "Ruby-MM-416 Digital Input (IO 303)" on page 5-34
- "Ruby-MM-416 Digital Output (IO 303)" on page 5-35

Board Characteristics

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

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-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:

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

Wiring for IO 304 Test Dongle

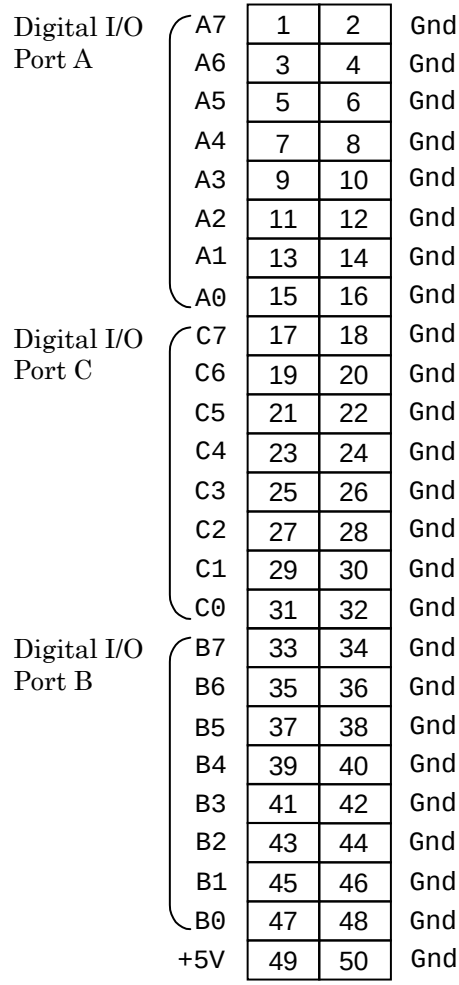
xPC TargetBox communicates with this board using two connectors. Some of the pins in the test dongles are connected together to allow for loop-back testing of this board:

- 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.



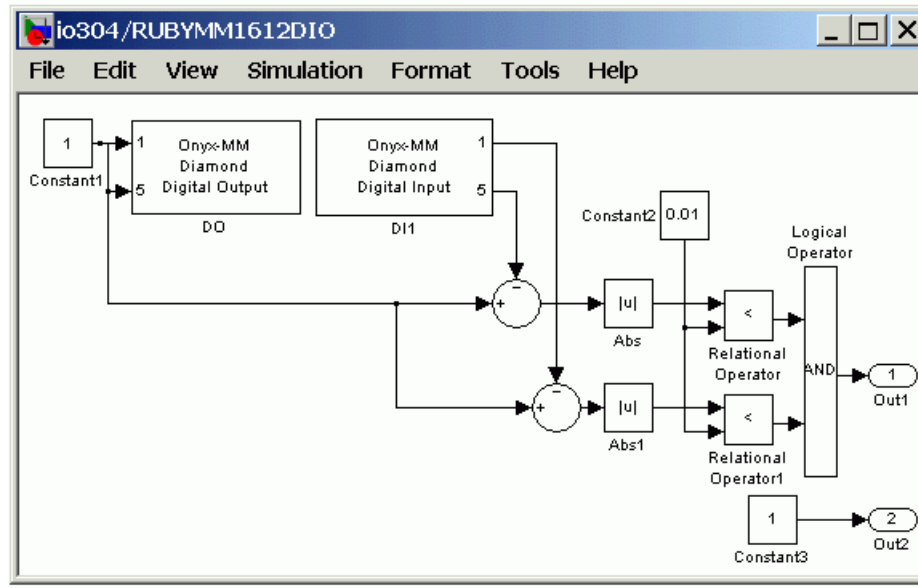
Testing Model IO 304

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 Outport block:

1 In the MATLAB Command Window, type

```
io304
```

The Simulink model for testing the IO 304 option opens.



Onyx-MM (IO 304) Driver Blocks

The Onyx-MM is an I/O board with 48 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:

- "Onyx-MM Digital Input (IO 304)"
- "Onyx-MM Digital Output (IO 304)"

Note xPC TargetBox does not support the counter/timer functionality of this board.

Board Characteristics

Board name	Onyx-MM
Manufacturer	Diamond Systems Corporation
Bus type	ISA (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. This section includes the following topics:

Wiring for IO 305 Test Dongle

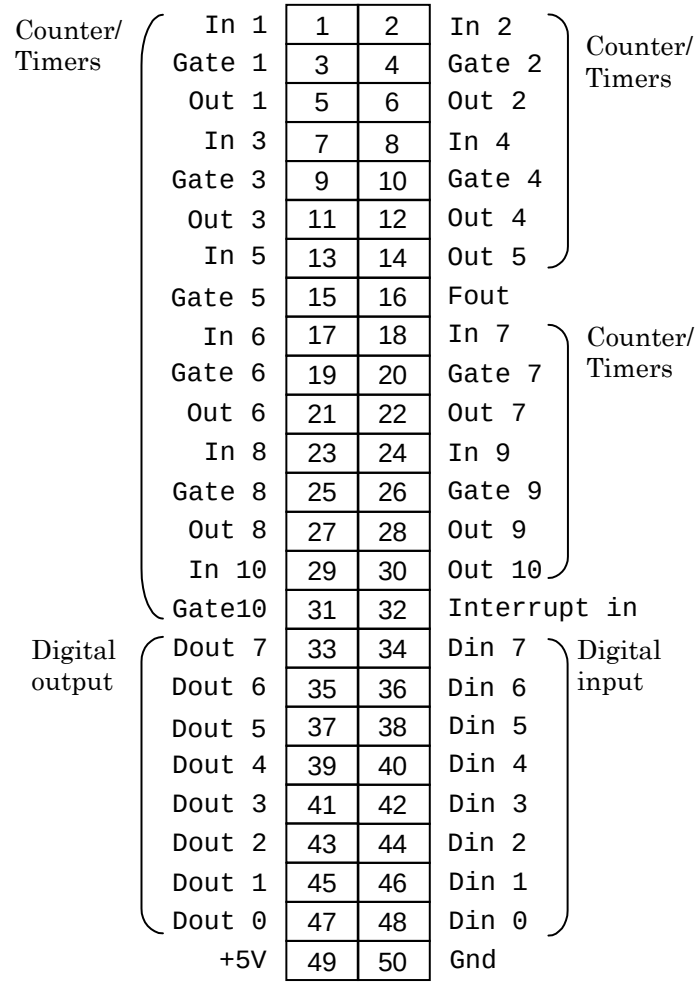
xPC TargetBox communicates with this board using one connector. Some of the pins in the test dongles are connected together to allow for loop-back testing of this board.

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.



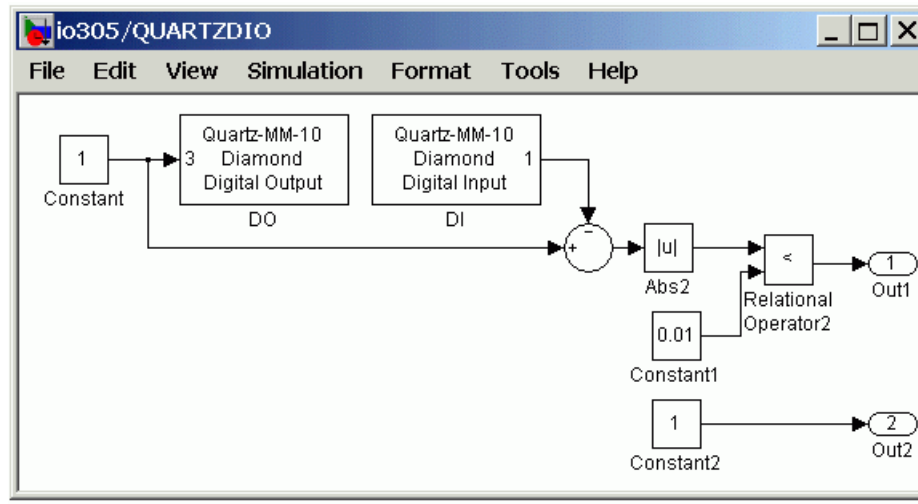
Testing Model IO 305

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 Outport 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)”
- “Quartz-MM-10 Digital Output (IO 305)”
- “Quartz-MM-10 Counter PWM (IO 305)”
- “Quartz-MM-10 Counter PWM & ARM (IO 305)”
- “Quartz-MM-10 Counter FM (IO 305)”

- “Quartz-MM-10 Counter FM & ARM (IO 305)”
- “Quartz-MM-10 PWM Capture (IO 305)”
- “Quartz-MM-10 FM Capture (IO 305)”
- “Quartz-MM (IO 305)”

Board Characteristics

Board name	Quartz-MM-10
Manufacturer	Diamond Systems Corporation
Bus type	ISA (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=1MHz`, `F2=100kHz`, `F3=10kHz`, `F4=1kHz`, or `F5=100Hz` to set the base frequency. The XTAL frequency is assumed to be 1 MHz; therefore the jumper on the Quarts-MM-10 must be in position `1` MHz, not `5` MHz.

Relative output frequency — Enter a value between 0 and 1. 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=100kHz` as the **Frequency base** and enter `0.175` as the **Relative output frequency**. 100 kHz x 0.175 = 17.5 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=1MHz`, `F2=100kHz`, `F3=10kHz`, `F4=1kHz`, or `F5=100Hz` to set the base frequency. The XTAL frequency is

assumed to be 1 MHz; therefore the jumper on the Quarts-MM-10 must be in position 1 MHz, not 5 MHz.

Relative output frequency — Enter a value less than 1. 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=100kHz` as the **Frequency base** and enter `0.175` as the **Relative output frequency**. `100` kHz x `0.175` = `17.5` 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=1MHz`, `F2=100kHz`, `F3=10kHz`, `F4=1kHz`, or `F5=100Hz` to set the base frequency. XTAL frequency is assumed to be `1` MHz; therefore the jumper on the Quarts-MM-10 must be in position `1` MHz, not `5` MHz.

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.

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

Hardware Output	Block Input Data Type	Scaling
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=1MHz`, `F2=100kHz`, `F3=10kHz`, `F4=1kHz`, or `F5=100Hz` to set the base frequency. The XTAL frequency is assumed to be `1` MHz; therefore the jumper on the Quarts-MM-10 must be in position `1` MHz, not `5` MHz.

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.

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=1MHz`, `F2=100kHz`, `F3=10kHz`, `F4=1kHz`, or `F5=100Hz` to set the base frequency. The XTAL frequency is assumed to be `1` MHz; therefore the jumper on the Quartz-MM-10 must be in position `1` MHz, not `5` MHz.

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. The CLK pin 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`, `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=1MHz`, `F2=100kHz`, `F3=10kHz`, `F4=1kHz`, or `F5=100Hz` to set the base frequency. The XTAL frequency is assumed to be `1` MHz; therefore the jumper on the Quartz-MM-10 must be in position `1` MHz, not `5` MHz.

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:

Note xPC TargetBox does not support the counter/timers on this board.

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 loop-back testing of this board.

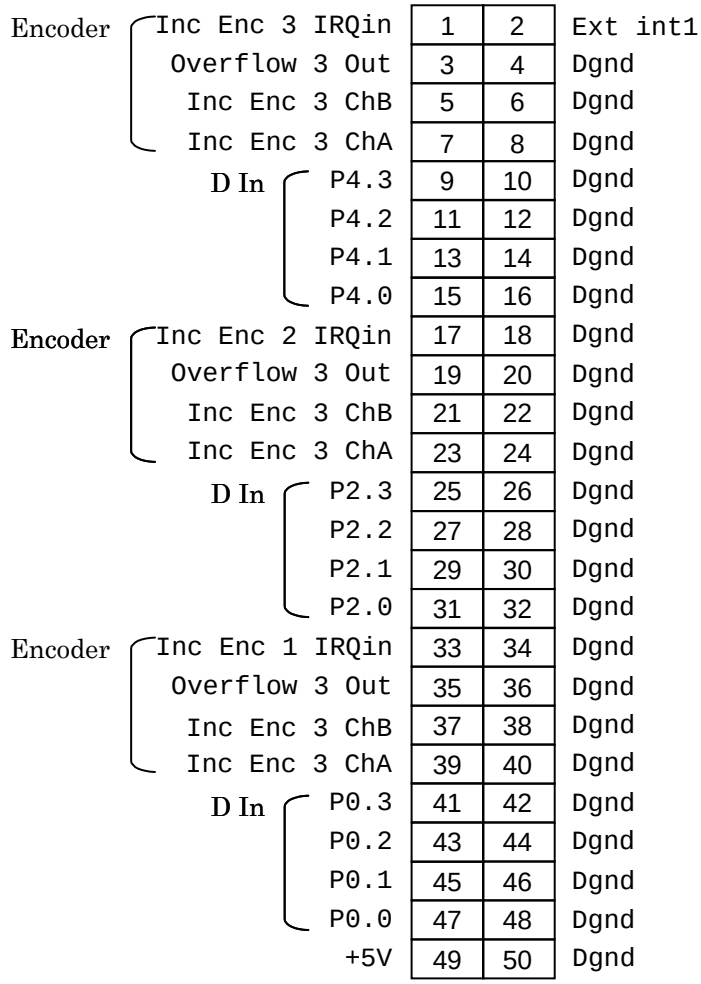
Header (P2) dongle — The following pins are connected together: 5–13, 7–15, 21–29, 23–31, 37–45, and 39–47.

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 loop-back 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/manuals/DM/dm5810.pdf`.

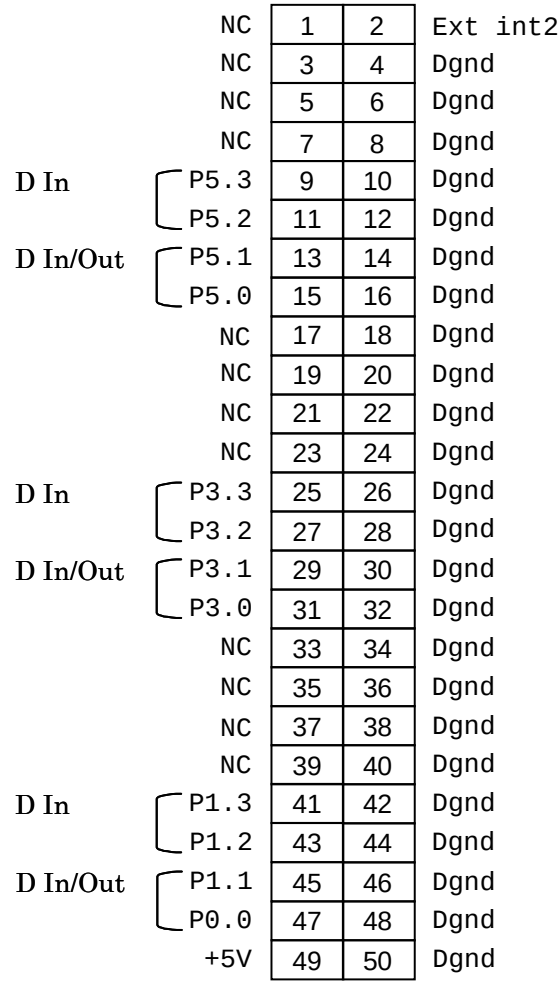
The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the figure below.



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.

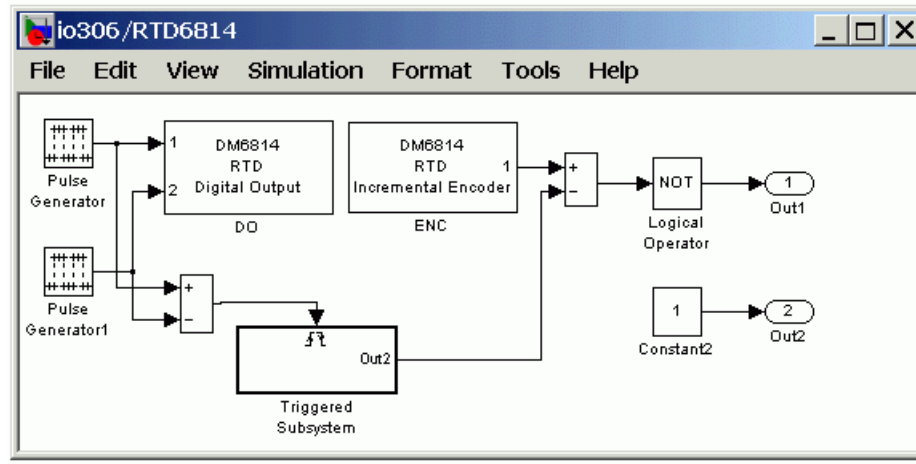


Testing Model IO 306

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)"
- "DM6814 Digital Input (IO 306)"
- "DM6814 Digital Output (IO 306)"

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)

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	`double`	counts

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.

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

Hardware Output	Block Input Data Type	Scaling
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

xPC TargetBox communicates with this board using two CAN serial connectors.

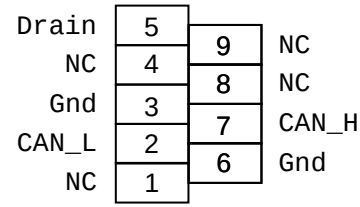
The test dongle is a CAN serial cable, with termination resistors, connected between the two CAN 9-pin connectors.

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.



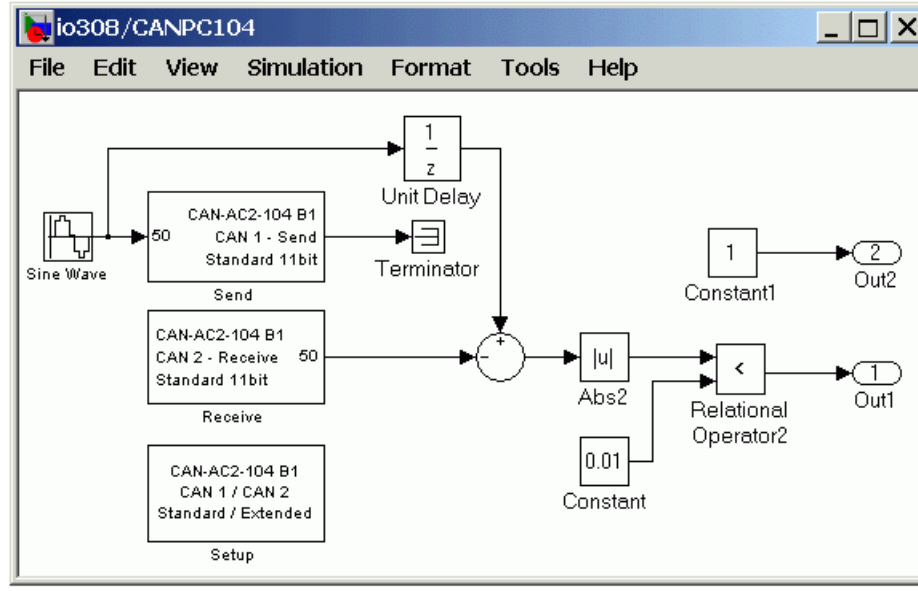
Testing Model IO 308

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 Outport block:

1 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 the xPC Target I/O Reference documentation.

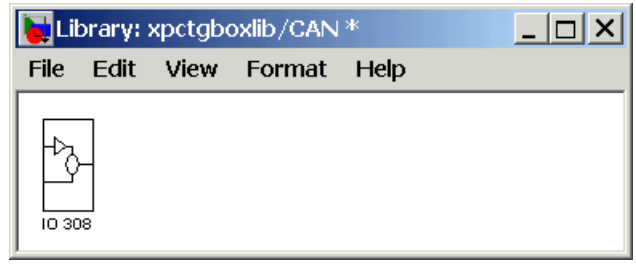
1 In the MATLAB Command Window, type

```
xpctgboxlib
```

The xPC TargetBox library dialog box opens.

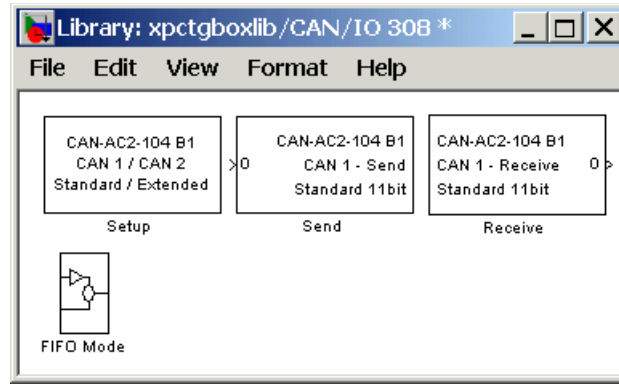
2 Click the block labeled **CAN**.

The CAN library for the xPC TargetBox opens. Currently there is only one supported CAN board for xPC TargetBox.



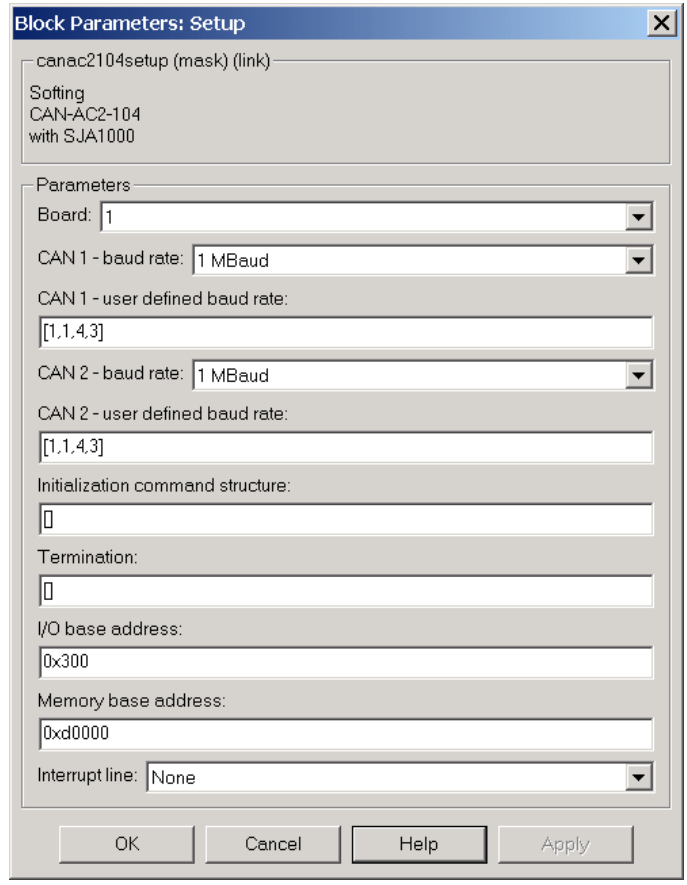
3 Click the block labeled **IO 308**.

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.



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 **I/O base address** 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 1 - baud rate — The fourth control (pop-up menu) lets you define 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-Segement-1,
Time-Segment-2 ]
```

For more information about these values, see the Softing user manual for this board.

Initialization and **Termination** — The sixth and seventh control, (edit fields) can be used to define CAN messages sent during initialization and termination of the Setup block.

I/O base address — The eighth control (edit field) is used to define 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 — The ninth control (edit field) is used to define 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 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

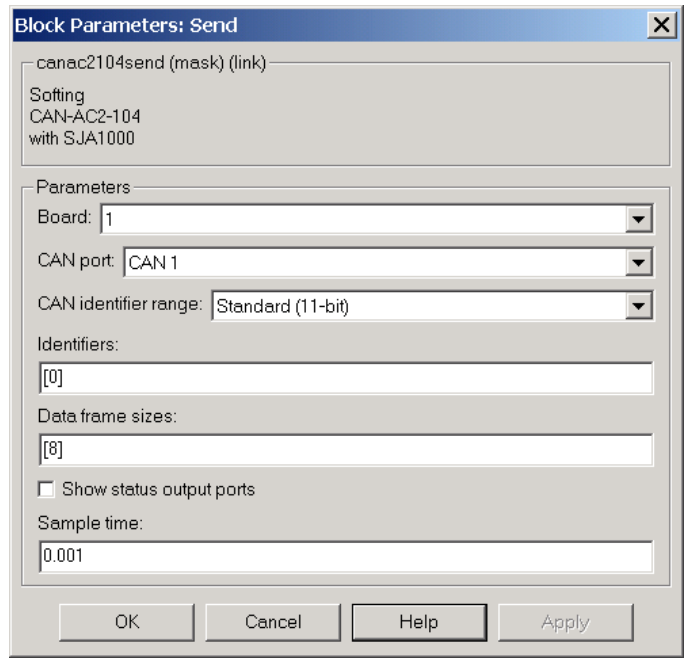
subset of the address range between 640 kilobytes and 1 MB for memory mapped devices, the address ranges must lie within the following range:

```
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 loop-back 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 — Select the CAN port to send the CAN message out.

CAN identifier range — The third control (pop-up menu) is used to select 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 — The fourth control (edit field) is used to define 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 — The fifth control (edit field) is used to define 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 have to be the same.

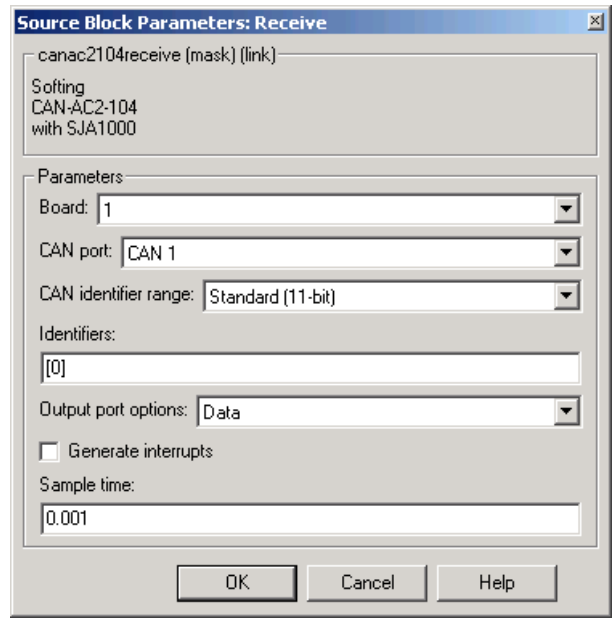
Show status output ports — The sixth control (check box) lets you enable status output ports for each identifier (CAN message). If the check box is selected, the block shows as many output ports as input ports. The data type of each output port is a `double` and the value is identical to the return argument of function `CANPC_write_object(...)`, described in the Softing user manual. Refer to the manual for more information.

Sample time — The seventh control (edit field) 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 — From the list, specify the existing 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 — From the list, select the CAN port from which to send the CAN message out.

CAN identifier range — From the list, select 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 — Specify 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 — From the list, define 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 — This check box lets you define 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 Send 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 Chapter 5, "CAN I/O Support for FIFO," in the xPC Target I/O Reference documentation.

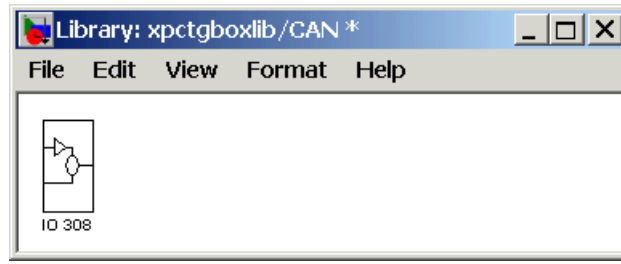
1 In the MATLAB Command Window, type

```
xpctgboxlib
```

The xPC TargetBox library dialog box opens.

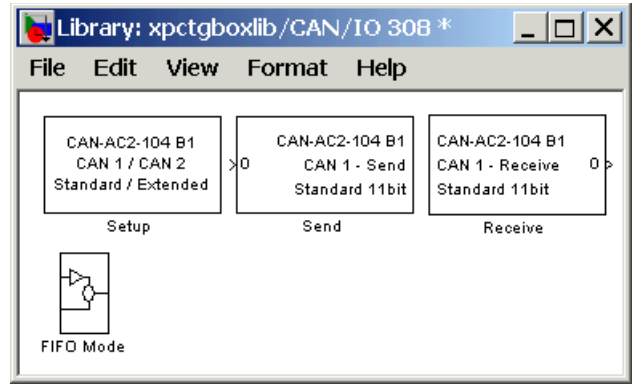
2 Click the block labeled CAN.

The CAN library for the xPC TargetBox opens. Currently there is only one supported CAN board for xPC TargetBox.



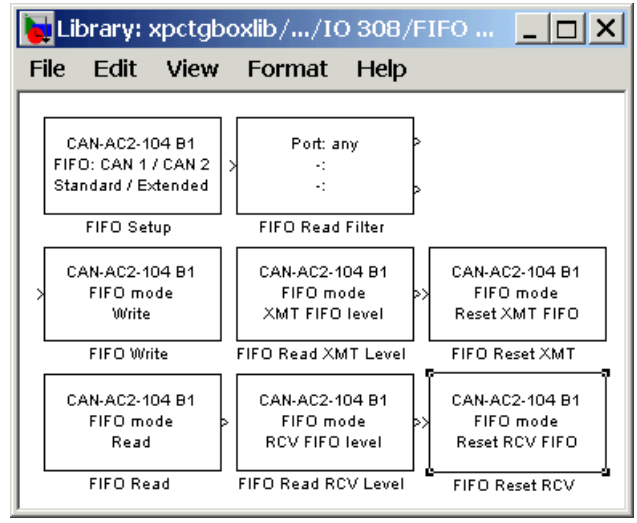
3 Click the block labeled IO 308.

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.

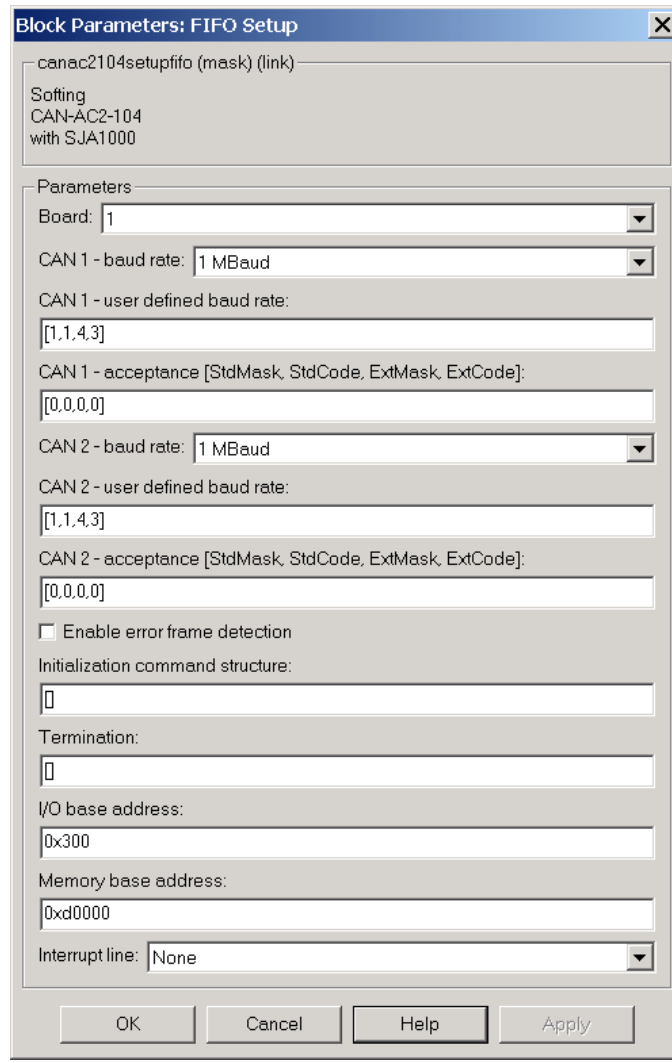
The **CAN / IO 308/ FIFO** dialog box opens with the driver blocks available for FIFO Mode CAN.



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 Chapter 5, “CAN I/O Support for FIFO,” in the xPC Target I/O Reference documentation.

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 Chapter 5, "CAN I/O Support for FIFO," in the xPC Target I/O Reference documentation.

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 Chapter 5, "CAN I/O Support for FIFO," in the xPC Target I/O Reference documentation.

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 bytes. 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 lie 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 loop-back 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.

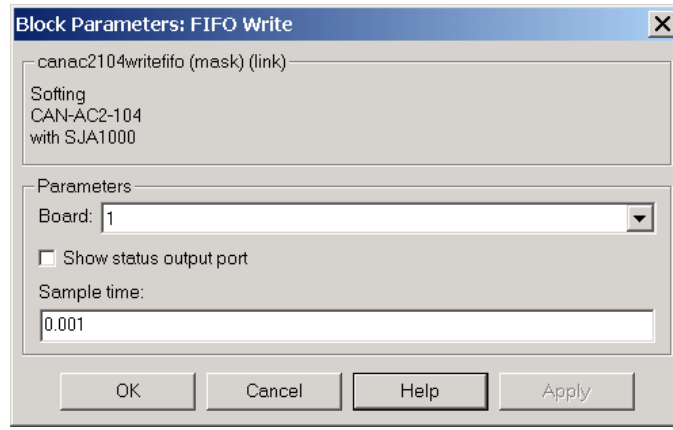
Data — This is the data for the data frame itself. It is defined as a `double` value (8 bytes). The CAN packing block is used to construct the data as a `double` value.

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*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 Chapter 5, "CAN I/O Support for FIFO," in the xPC Target I/O Reference documentation.

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*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*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 Chapter 5, "CAN I/O Support for FIFO," in the xPC Target I/O Reference documentation.

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.

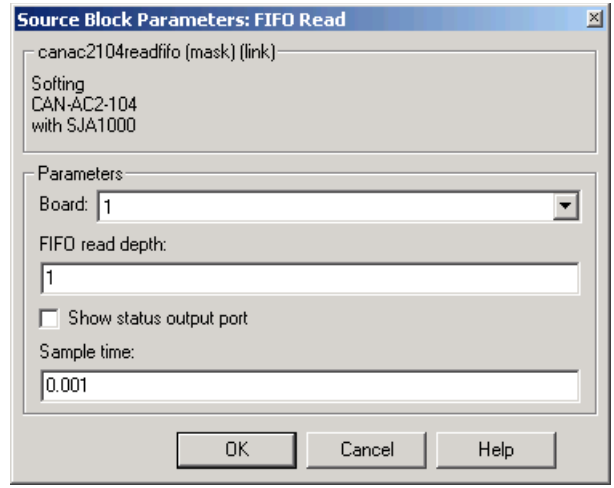
Event type — This value defines the type of event read from the receive FIFO. The following values are defined from the Softing user manual:

```
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 μs.

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*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.

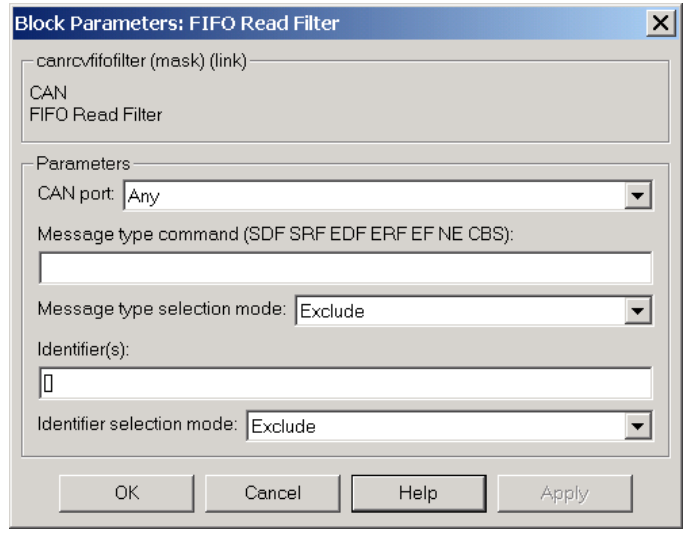
Note Only use one instance of this block per physical CAN board in your model. Otherwise, you might get unwanted behavior where one instance reads events while the events are being processed by blocks connected to the other, second instance.

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*6`. The two output ports are of type `double` as well. The first output is a row vector (`1*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.

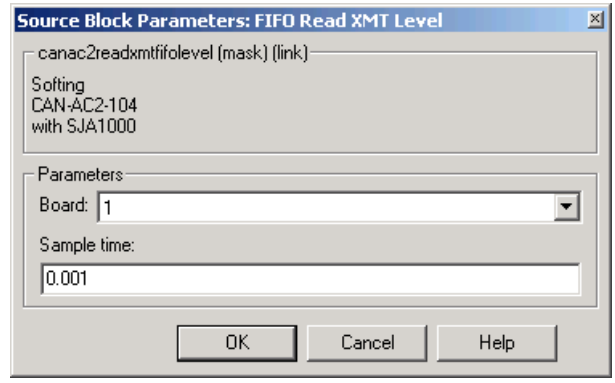
Identifier selection mode — Defines how the identifier criterion entered in the control above is treated. If you select `Include`, the identifier criterion is the sum of all specified identifiers. If you select `Exclude`, the identifier criterion is equal to all identifiers minus the specified identifiers.

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 Chapter 5, "CAN I/O Support for FIFO," in the xPC Target I/O Reference documentation.

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:

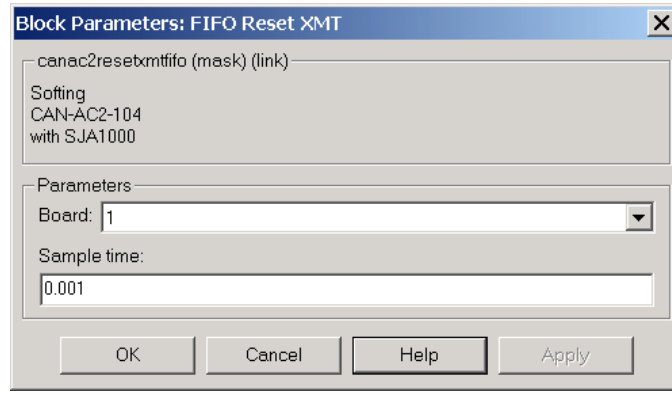
Board — Defines which board to access to read the current transmit FIFO level. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

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:

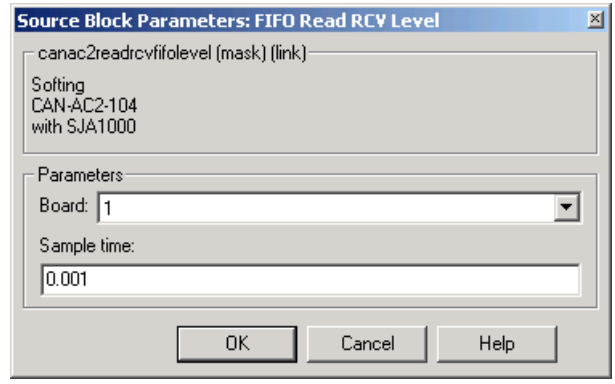
Board — Defines which board to access to reset the transmit FIFO. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

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:

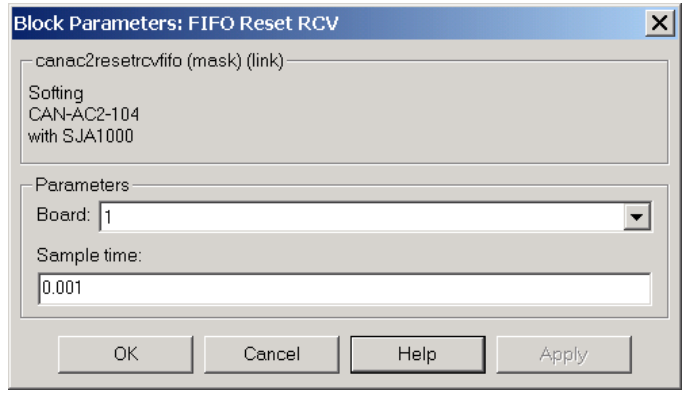
Board — Defines which board to access to read the current receive FIFO level. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

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.

Index

E

F

H

I

J

K

L

M

N

P

Q

R

S

T

W

X