

# xPC Target

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

- Modeling
- Simulation
- Implementation

I/O Reference

*Version 3*



## How to Contact The MathWorks:



www.mathworks.com  
comp.soft-sys.matlab  
www.mathworks.com/contact\_TS.html

Web  
Newsgroup  
Technical Support



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

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



508-647-7000 (Phone)



508-647-7001 (Fax)



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

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

*xPC Target I/O Reference*

© COPYRIGHT 2000–2006 by The MathWorks, Inc.

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

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

### Trademarks

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

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

### Patents

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

## Revision History

November 2000	Online only	Revised for Version 1.1 (Release 12)
June 2001	Online only	Revised for Version 1.2 (Release 12.1)
September 2001	Online only	Revised for Version 1.3 (Release 12.1+)
July 2002	Online only	Revised for Version 2 (Release 13)
September 2002	Online only	Revised for Version 2.0.1 (Release 13)
September 2003	Online only	Revised for Version 2.0.1 (Release 13SPI)
June 2004	Online only	Revised for Version 2.5 (Release 14)
August 2004	Online only	Revised for Version 2.6 (Release 14+)
October 2004	Online only	Revised for Version 2.6.1 (Release 14SP1)
November 2004	Online only	Revised for Version 2.7 (Release 14SP1+)
March 2005	Online only	Revised for Version 2.7.2 (Release 14SP2)
September 2005	Online only	Revised for Version 2.8 (Release 14SP3)
March 2006	Online only	Revised for Version 2.9 (Release 2006a)
May 2006	Online only	Revised for Version 3.0 (Release 2006a+)



## xPC Target I/O Library

**1**

<b>I/O Driver Blocks</b> .....	<b>1-2</b>
I/O Driver Block Library .....	<b>1-2</b>
Memory-Mapped Devices .....	<b>1-5</b>
ISA Bus I/O Devices .....	<b>1-6</b>
PCI Bus I/O Devices .....	<b>1-6</b>
xPC Target I/O Driver Structures .....	<b>1-7</b>
PWM and FM Driver Block Notes .....	<b>1-9</b>
Updated Driver Information .....	<b>1-10</b>
<b>Adding I/O Blocks with the xPC Target Library</b> .....	<b>1-11</b>
<b>Adding I/O Blocks with the Simulink Library</b>	
<b>Browser</b> .....	<b>1-15</b>
<b>Defining I/O Block Parameters</b> .....	<b>1-20</b>

## Serial Communications Support

**2**

<b>Introduction to Serial Drivers</b> .....	<b>2-2</b>
Hardware Connections for RS-232 .....	<b>2-3</b>
Host and Target PC Communication .....	<b>2-4</b>
<b>xPC Target RS-232 and 422/485 Drivers (Composite)</b> ..	<b>2-5</b>
Adding RS-232 Blocks .....	<b>2-6</b>
Building and Running the Target Application	
(Composite) .....	<b>2-12</b>
RS-232/422/485 Simulink Block Reference .....	<b>2-13</b>
<b>xPC Target RS-232 Drivers (Conventional)</b> .....	<b>2-19</b>

Simulink Blocks for RS-232 I/O (Conventional) .....	2-19
MATLAB Message Structures for RS-232 I/O (Conventional) .....	2-20
RS-232 Synchronous Mode (Conventional) .....	2-21
RS-232 Asynchronous Mode (Conventional) .....	2-31
RS-232 Simulink Block Reference (Conventional) .....	2-44
RS-232 MATLAB Structure Reference (Conventional) ....	2-44
RS-232 Binary Mode (Conventional) .....	2-48
 <b>Boards and Blocks — Alphabetical List</b> .....	 2-53

## GPIB I/O Support

# 3

<b>Introduction to GPIB Drivers</b> .....	<b>3-2</b>
Hardware Connections for GPIB .....	3-2
Simulink Blocks for GPIB .....	3-3
MATLAB Message Structures for GPIB .....	3-3
 <b>Using GPIB Drivers</b> .....	 3-5
Adding GPIB Driver Blocks .....	3-5
Creating GPIB Message Structures .....	3-9
 <b>GPIB MATLAB Structure Reference</b> .....	 3-13
GPIB Initialization and Termination Message Structures .....	3-13
GPIB Send/Receive Message Structure .....	3-14
Shortcuts and Features for Messages .....	3-17
Supported Data Types for Message Fields .....	3-19
 <b>Boards and Blocks — Alphabetical List</b> .....	 3-20

<b>Introduction</b> .....	<b>4-3</b>
xPC Target CAN Library .....	<b>4-3</b>
CAN-AC2 .....	<b>4-5</b>
CAN-AC2-PCI .....	<b>4-5</b>
CAN-AC2-104 .....	<b>4-6</b>
<b>Model Execution Driven by CAN Messages (Interrupt Capability of CAN Receive Blocks)</b> .....	<b>4-7</b>
CAN-AC2 (ISA) .....	<b>4-7</b>
CAN-AC2-PCI .....	<b>4-8</b>
CAN-AC2-104 (PC/104) .....	<b>4-9</b>
<b>Defining Initialization and Termination CAN Messages</b> .....	<b>4-11</b>
Example .....	<b>4-12</b>
<b>CAN-AC2 and CANopen Devices</b> .....	<b>4-14</b>
<b>Constructing and Extracting CAN Data Frames</b> .....	<b>4-15</b>
<b>Detecting Time-Outs When Receiving CAN Messages</b> ..	<b>4-16</b>
<b>CAN Blocks for the CAN-AC2 (ISA) with Philips PCA 82C200 CAN Controller</b> .....	<b>4-17</b>
<b>CAN Blocks for the CAN-AC2 (ISA) with Intel 82527 CAN Controller</b> .....	<b>4-26</b>
<b>CAN Blocks for the CAN-AC2-PCI with Philips SJA1000 CAN Controller</b> .....	<b>4-36</b>
<b>CAN Blocks for the CAN-AC2-104 (PC/104) with Philips SJA1000 CAN Controller</b> .....	<b>4-46</b>

## CAN I/O Support for FIFO

### 5

<b>Introduction</b> .....	<b>5-2</b>
FIFO Mode Drivers for CAN Boards from Softing .....	<b>5-3</b>
<b>Acceptance Filters</b> .....	<b>5-6</b>
Examples .....	<b>5-7</b>
<b>Examples</b> .....	<b>5-8</b>
Example 1 .....	<b>5-8</b>
Example 2 .....	<b>5-11</b>
Example 3 .....	<b>5-12</b>
Example 4 .....	<b>5-13</b>
Example 5 .....	<b>5-14</b>
Example 6 .....	<b>5-14</b>
<b>CAN FIFO Blocks for the CAN-AC2-PCI with Philips SJA1000 CAN Controller</b> .....	<b>5-17</b>
<b>CAN FIFO Blocks for the CAN-AC2-104 with Philips SJA1000 CAN Controller</b> .....	<b>5-39</b>

## UDP I/O Support

### 6

<b>User Datagram Protocol (UDP)</b> .....	<b>6-2</b>
What Is UDP? .....	<b>6-2</b>
Why UDP? .....	<b>6-4</b>
Note on UDP Communication .....	<b>6-4</b>
<b>xPC Target UDP Example</b> .....	<b>6-5</b>
<b>UDP Communication Setup</b> .....	<b>6-11</b>
<b>Boards and Blocks — Alphabetical List</b> .....	<b>6-13</b>



## ARINC 429 Support

### 7

<b>Boards and Blocks — Alphabetical List</b> .....	<b>7-2</b>
--	------------

## MIL-STD-1553 Support

### 8

<b>Introduction</b> .....	<b>8-2</b>
Before You Start .....	<b>8-2</b>
Remote Terminal Operation .....	<b>8-5</b>
Bus Controller Operation .....	<b>8-7</b>
Remote Terminal and Bus Controller Operation .....	<b>8-9</b>
Bus Monitor Operation .....	<b>8-10</b>
<b>Boards and Blocks — Alphabetical List</b> .....	<b>8-13</b>

## Access IO

### 9

<b>Boards and Blocks</b> .....	<b>9-2</b>
--------------------------------	------------

## ADDI-DATA

### 10

<b>Boards and Blocks — Alphabetical List</b> .....	<b>10-2</b>
--	-------------

**Adlink**

**11**

**Boards and Blocks — Alphabetical List** ..... 11-2

**Advantech**

**12**

**Boards and Blocks — Alphabetical List** ..... 12-3

**Analogic**

**13**

**Boards and Blocks — Alphabetical List** ..... 13-2

**BittWare**

**14**

**Running Models with BittWare Blocks** ..... 14-2

- Model Notes ..... 14-2
- Frame Size, Sample Rate, and Sampletime Parameter  
Notes ..... 14-4

**Boards and Blocks — Alphabetical List** ..... 14-6

**BVM**

**15**

**Boards and Blocks** ..... 15-2

**Commtech**

**16**

**Condor**

**17**

**Contec**

**18**

**Boards and Blocks — Alphabetical List . . . . . 18-3**

**Data Translation**

**19**

**Boards and Blocks — Alphabetical List . . . . . 19-4**

**Diamond**

**20**

**Boards and Blocks — Alphabetical List . . . . . 20-3**

**General Standards**

**21**

**Overview of PMC-ADADIO Functionality . . . . . 21-2**  
A/D Blocks . . . . . 21-3  
Create Enable Signal Blocks . . . . . 21-5  
D/A Blocks . . . . . 21-7

Interleaving Analog Input and Analog Output Blocks . . . .	21-10
Using Multiple Boards for Simultaneous Analog to Digital Conversion . . . . .	21-12

<b>Boards and Blocks — Alphabetical List . . . . .</b>	<b>21-14</b>
--	--------------

**Humusoft**

**22** |

<b>Boards and Blocks . . . . .</b>	<b>22-2</b>
------------------------------------	-------------

**Keithley**

**23** |

<b>Boards and Blocks — Alphabetical List . . . . .</b>	<b>23-2</b>
--	-------------

**Measurement Computing (Formerly Computer  
Boards)**

**24** |

<b>Boards and Blocks — Alphabetical List . . . . .</b>	<b>24-6</b>
--	-------------

**MPL**

**25** |

<b>Boards and Blocks . . . . .</b>	<b>25-2</b>
------------------------------------	-------------

**National Instruments**

---

**26**

**Boards and Blocks — Alphabetical List . . . . . 26-5**

**North Atlantic Industries, Inc. (Formerly Apex)**

---

**27**

**Boards and Blocks — Alphabetical List . . . . . 27-3**

**Quanser**

---

**28**

**Boards and Blocks . . . . . 28-2**

**Quatech**

---

**29**

**Real Time Devices**

---

**30**

**Boards and Blocks — Alphabetical List . . . . . 30-3**

**SBS Technologies**

---

**31**

**Broadcast Memory . . . . . 31-3**

Create Shared Memory Partitions .....	31-3
Initialize Shared Memory Nodes .....	31-5
<b>SBS Shared Memory Structure Reference .....</b>	<b>31-8</b>
Shared Memory Partition Structure .....	31-8
Shared Memory Node Initialization Structure .....	31-10
<b>Boards and Blocks — Alphabetical List .....</b>	<b>31-14</b>

## Sensoray

### 32

<b>Boards and Blocks — Alphabetical List .....</b>	<b>32-2</b>
--	-------------

## Softing

### 33

## Systran

### 34

<b>Before You Start .....</b>	<b>34-2</b>
Create Shared Memory Partitions .....	34-2
Initialize Shared Memory Nodes .....	34-4
<b>Systran Shared Memory Structure Reference .....</b>	<b>34-6</b>
Shared Memory Partition Structure .....	34-6
Shared Memory Node Initialization Structure .....	34-10
<b>Boards and Blocks — Alphabetical List .....</b>	<b>34-16</b>

## United Electronic Industries (UEI)

### 35

<b>Grouping the UEI Boards</b> .....	<b>35-5</b>
Changing the Board Associated with the Block .....	<b>35-5</b>
Getting Information on a Specific Board .....	<b>35-6</b>
<b>Analog Input Frame Driver Blocks</b> .....	<b>35-7</b>
Notes on Master and Slave Boards .....	<b>35-7</b>
Interrupt Numbers .....	<b>35-8</b>
Interrupt Configuration .....	<b>35-10</b>
Example Models .....	<b>35-12</b>
<b>Boards and Blocks — Alphabetical List</b> .....	<b>35-14</b>

## xPC Target Support for Vector CANape

### 36

<b>Vector CANape</b> .....	<b>36-2</b>
Notes on xPC Target and Vector CANape .....	<b>36-3</b>
<b>Configuring xPC Target and Vector CANape</b> .....	<b>36-5</b>
Getting Started .....	<b>36-5</b>
Creating a New Vector CANape Project to Associate with a Particular Target Application .....	<b>36-7</b>
Associating an Existing Vector CANape Project with a Particular Target PC .....	<b>36-9</b>
Providing A2L Files for the Vector CANape Database ....	<b>36-10</b>

## Versallogic

### 37

<b>Boards and Blocks</b> .....	<b>37-2</b>
--------------------------------	-------------

## 38

<b>Before You Start</b> .....	38-2
Create Shared Memory Partitions .....	38-2
Initialize Shared Memory Nodes .....	38-4
<b>VMIC Shared Memory Structure Reference</b> .....	38-6
Shared Memory Partition Structure .....	38-6
Shared Memory Node Initialization Structure .....	38-8
<b>Boards and Blocks — Alphabetical List</b> .....	38-13

### Miscellaneous Blocks

## 39

<b>Asynchronous Event Support</b> .....	39-2
Adding an Asynchronous Event .....	39-2
Asynchronous Interrupt Examples .....	39-5
<b>Blocks — Alphabetical List</b> .....	39-6

### Utility Blocks

## 40

<b>Blocks — Alphabetical List</b> .....	40-2
---	------

### Obsolete Drivers

## 41

<b>xPC Target Library of Obsolete Drivers</b> .....	41-2
---	------



**Blocks — Alphabetical List** ..... **41-3**



# xPC Target I/O Library

---

xPC Target is a solution for prototyping, testing, and deploying real-time systems using standard PC hardware. In support of this, xPC Target allows you to add I/O blocks to your model. I/O blocks in xPC Target provide a particular function of an I/O board. By using I/O blocks in your model, you can generate executable code tuned specifically for your hardware. This chapter includes the following sections:

I/O Driver Blocks (p. 1-2)

Introduction to the xPC Target I/O library of driver blocks

Adding I/O Blocks with the xPC Target Library (p. 1-11)

Adding I/O blocks to a model using the xPC Target library

Adding I/O Blocks with the Simulink Library Browser (p. 1-15)

Adding I/O blocks to a model using the Simulink® library browser

Defining I/O Block Parameters (p. 1-20)

Configuring I/O driver blocks

## I/O Driver Blocks

You add I/O driver blocks to your Simulink model to connect your model to physical I/O boards. These I/O boards then connect to the sensors and actuators in the physical system. This section includes the following topics:

- “I/O Driver Block Library” on page 1-2
- “Memory-Mapped Devices” on page 1-5
- “PCI Bus I/O Devices” on page 1-6
- “xPC Target I/O Driver Structures” on page 1-7
- “Updated Driver Information” on page 1-10

Refer to the following sections for descriptions on how to add I/O blocks to your model, and how to configure those blocks:

- “Adding I/O Blocks with the xPC Target Library” on page 1-11
- “Adding I/O Blocks with the Simulink Library Browser” on page 1-15
- “Defining I/O Block Parameters” on page 1-20

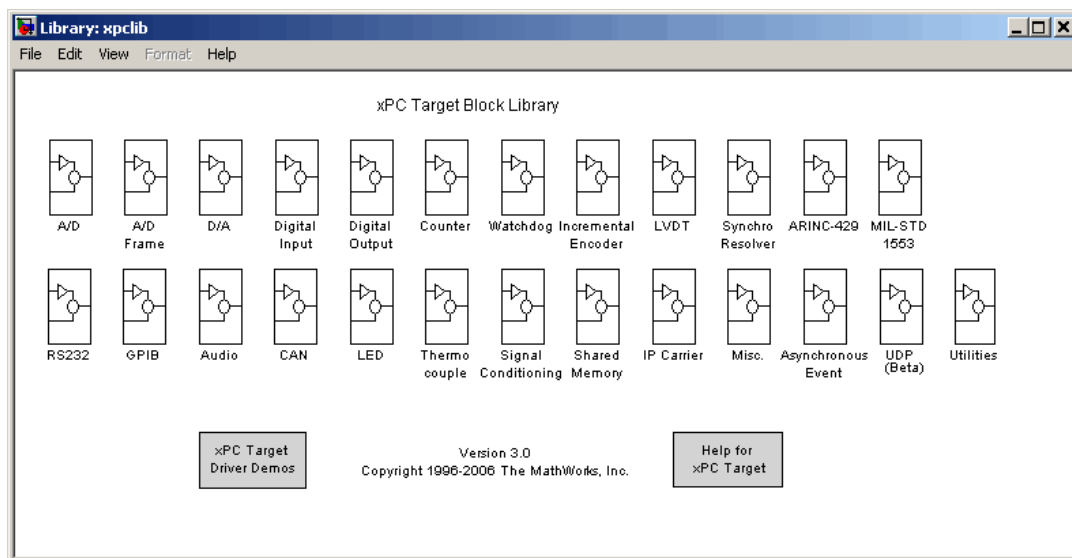
### I/O Driver Block Library

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

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

You can open the I/O device driver library with the MATLAB® command `xpclib`. The library `xpclib` contains sublibraries grouped by the type of I/O function they provide.

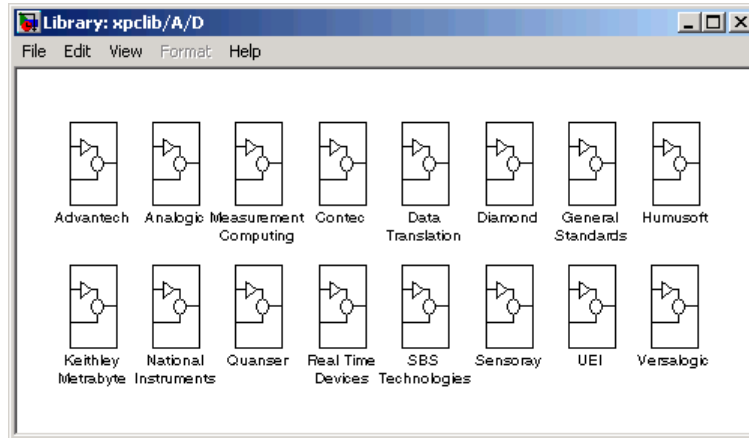
**Note** Opening a dialog box for a source block causes Simulink to pause. While Simulink is paused, you can edit the parameter values. You must close the dialog box to have the changes take effect and allow Simulink to continue. In particular, hardware input blocks in the xPC Target library (blocks that acquire data from hardware) are affected by this change.



This window also contains the following blocks:

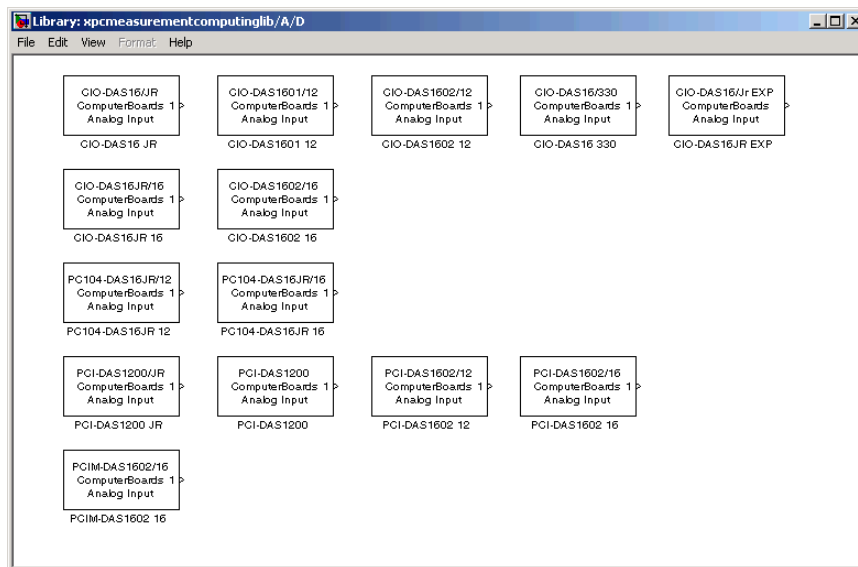
- **xPC Target Driver Demos** — When you double-click this block, the sublibrary opens, displaying a list of xPC Target driver demo groups.
- **Help for xPC Target** — When you double-click this block, the xPC Target roadmap page is displayed. You can access the xPC Target documentation with this block.

When you double-click one of I/O block groups, the sublibrary opens, displaying a list grouped by manufacturer as shown below.



Double-clicking one of the manufacturer groups then displays the set of I/O device driver blocks for the specified I/O functionality (for example, A/D, D/A, Digital Inputs, Digital Outputs, and so on).

The following figure shows the A/D drivers for the manufacturer Measurement Computing, Inc.



When you double-click one of these blocks, a Block Parameters dialog box opens, allowing you to enter hardware-specific parameters. Parameters typically include

- Sample time
- Number of channels
- Voltage range
- PCI slot (PCI boards)
- Base address (ISA boards)

## Memory-Mapped Devices

Some supported boards in the xPC Target I/O library are memory-mapped devices, for example, Burr-Brown boards. These memory-mapped boards are accessed in the address space between 640 K and 1 MB in the lower memory

area. xPC Target reserves a 112 KB memory space for memory-mapped devices in the address range

C0000 - DBFFF

Some drivers for memory-mapped devices allow you to select an address range supported by the device, but not supported by xPC Target. For example, the CAN drivers for Softing allow you to select memory ranges above DBFFF. Base addresses of memory-mapped devices must be chosen within this memory space for your target application to work properly. Select a memory range supported by both the device and xPC Target.

## **ISA Bus I/O Devices**

There are two types of ISA boards:

- Jumper addressable ISA cards
- PnP (Plug and Play) ISA cards

xPC Target only supports jumper addressable ISA cards (non-PnP ISA boards) where you have to set the base address manually.

## **PCI Bus I/O Devices**

The xPC Target I/O library supports I/O boards with a PCI bus. During the boot process, the BIOS creates a conflict-free configuration of base addresses and interrupt lines for all PCI devices in the target system. The user does not need to define any base address information in the dialog boxes of the drivers.

All PCI device driver blocks have an additional entry in their dialog boxes. This entry is called `PCI Slot (-1 Autodetect)` and allows you to use several identical PCI boards within one target system. This entry uses a default value of `-1`, which allows the driver to search the entire PCI bus to find the board. If you specify a single number, `X`, greater than `0`, the driver uses the board in bus `0`, slot `X`. When more than one board of the same type is found, you must use a designated slot number and avoid the use of autodetection. For manually setting the slot number you use a number greater than or equal to `0`. If the board is not able to locate this slot in the target PC, your target application will generate an error message after downloading.



If this additional entry is set to any value equal to or greater than 0, you must be aware of the manufacturer's identification number (Vendor ID) and the board identification number (Device ID) of those boards supported by the I/O library. When the target is booted, the BIOS is executed and the target PC monitor shows parameters for any PCI boards installed on the target PC. An example is shown below:

Bus No	Device No.	Func. No.	Vendor ID	Device ID	Device Class	IRQ
0	4	1	8086	7111	IDE controller	14/15
0	4	2	8086	7112	Serial bus controller	10
0	11	0	1307	000B	Unknown PCI device	N/A
1	0	0	12D2	0018	Display controller	11

In this example, the third line indicates the location of the Measurement Computing PCI-DIO48 board. This is known since the Measurement Computing vendor ID is 0x1307 and the device ID is 0xb. In this case, you now can see that the Measurement Computing board is plugged into PCI slot 11 (Device No.), and that this value must be entered in the dialog box entry in your I/O device driver for each model that uses this I/O device.

## xPC Target I/O Driver Structures

Properties for xPC Target I/O drivers are usually defined using the parameter dialog box associated with each Simulink block. However, for more advanced drivers, the available fields defined by text boxes, check boxes, and pull-down lists are inadequate to define the behavior of the driver. In such cases, a more textual description is needed to indicate what the driver has to do at runtime. *Textual* in this context refers to a programming-language-like syntax and style.

xPC Target currently uses a string description contained in message structures for the conventional RS-232, GPIB, CAN (initialization), and the general counter drivers (AMD9513).

**What is a message structure?** — A message structure is a MATLAB array with each cell containing one complete message (command). A message consists of one or more statements.

First message      Second message      Third message

Message(1).field	Message(1).field	Message(1).field
Message(1).field	Message(1).field	Message(1).field
Message(1).field	Message(1).field	Message(1).field

**Syntax of a message statement** — Each statement in a message has the following format:

```
Structure_name(index).field_name = <field string or value>
```

The field names are defined by the driver, and need to be entered with the correct upper- and lowercase letters. However, you can choose your own structure name and enter that name into the driver parameter dialog box.

**Creating a message structure** — You could enter the message structure directly in the edit field of the driver parameter dialog box. But because the message structure is an array and very large, this becomes cumbersome very easily.

A better way is to define the message structure as an array in an M-file and pass the structure array to the driver by referencing it by name. For example, to initialize an external A/D module and acquire a value during each sample interval, create an M-file with the following statements:

```
Message(1).senddata='InitADConv, Channel %d'
Message(1).inputports=[1]
Message(1).recdata=''
Message(1).outputports=[]

Message(2).senddata='Wait and Read converted Value'
Message(2).inputports=[]
Message(2).recdata='%f'
Message(2).outputports=[1]
```

This approach is different from other xPC Target driver blocks:

- The M-file containing the definition of the message structure has to be executed before the model is opened.

After creating your Simulink model and message M-file, set the preload function of the Simulink model to load the M-file the next time you open the model. In the MATLAB window, type

```
set_param(gcs, 'PreLoadFcn', 'M-file_name')
```

- When you move or copy the model file to a new directory, you also need to move or copy the M-file defining the message structure.

During each sample interval, the driver block locates the structure defined in the Block Parameters dialog box, interprets the series of messages, and executes the command defined by each message.

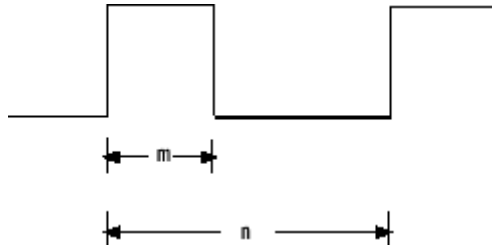
**Specific drivers and structures** — For detailed information on the fields in a message structure, see the following chapters in this document:

- Chapter 2, “Serial Communications Support”
- Chapter 3, “GPIB I/O Support”
- Chapter 4, “CAN I/O Support”

## PWM and FM Driver Block Notes

In PWM and FM driver blocks, your control over the output frequency and duty cycle is not always precise. In particular, these values are affected by the way that the base frequency is selected, as described in this section. The base frequency value is exact.

At the beginning of each sample time, two unsigned 16-bit integers,  $n$  and  $m$ , are computed based on the block parameters and the current values of the input signals. During the current sample period, the output signal is held high for  $m$  cycles of the base frequency, low for the next  $n-m$  cycles, high for the next  $m$  cycles, and so forth.



For a base frequency  $b$ , this results in a rectangular output signal of frequency  $b/n$  and duty cycle  $m/n$ . Because  $m$  and  $n$  must be integers, it is not possible to provide a continuous range of output frequencies and duty cycles with perfect exactness.

For example, assume that you want to configure an FM block with a duty cycle ( $m/n$ ) of  $1/2$ . The input signal  $f$  to this block is a relative frequency. It specifies an output frequency of  $b \times f$ . Because  $m$  and  $n$  must be integers, it is not always possible to find values of  $m$  and  $n$  such that  $f$  will equal  $b/n$  exactly and  $n$  will equal  $2 \times m$  (duty cycle  $m/n = 1/2$ ) exactly. Such an exact match is only possible when the input signal  $f$  equals  $1/4$ ,  $1/6$ ,  $1/8$ , and so forth. The output frequencies for the intervening input signal  $f$  values are approximate. The errors are smaller as  $f$  approaches 0 and larger as  $f$  approaches 1.

Hint, to achieve the smallest margin of error, select the largest possible base frequency. The fact that  $n$  and  $m$  must be 16-bit integers imposes a lower limit of  $b / (2^{16} - 1)$  on the frequencies that can be generated using a given base frequency.

## Updated Driver Information

Because new drivers are always being added, and existing drivers are always being updated, not all of the information about these drivers is included in the online or printed documentation.

For updated and additional driver information, see the developer Web site at

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

## Adding I/O Blocks with the xPC Target Library

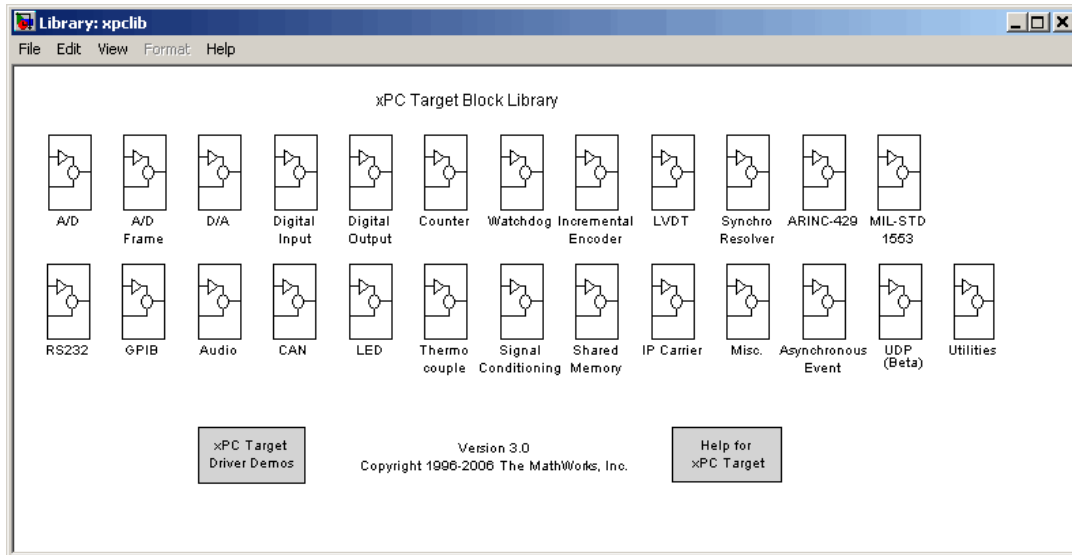
xPC Target includes a Simulink block library for I/O drivers. The highest hierarchical level in the library is grouped by I/O function. The second level is grouped by board manufacturer. The manufacturer groups within this second level contain the driver blocks for specific boards.

This procedure uses the Simulink model `xpc_osc.mdl` as an example of how to add and connect I/O blocks:

- 1 In the MATLAB window, type

```
xpclib
```

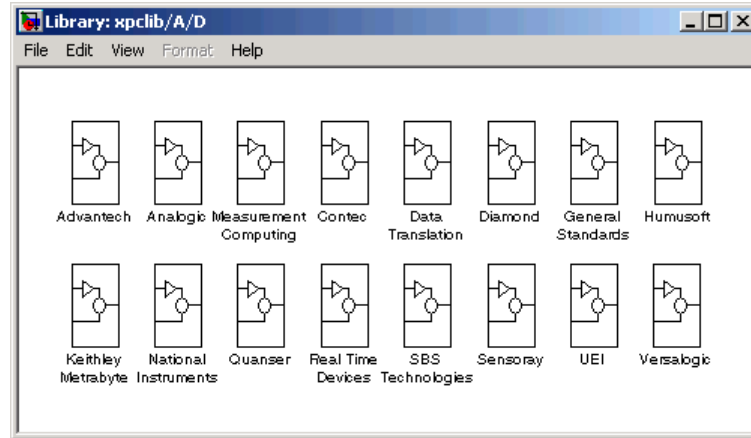
The **Library: xpclib** window opens.



Alternatively, you can access the I/O driver library with the Simulink Library Browser. See “Adding I/O Blocks with the Simulink Library Browser” on page 1-15.

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

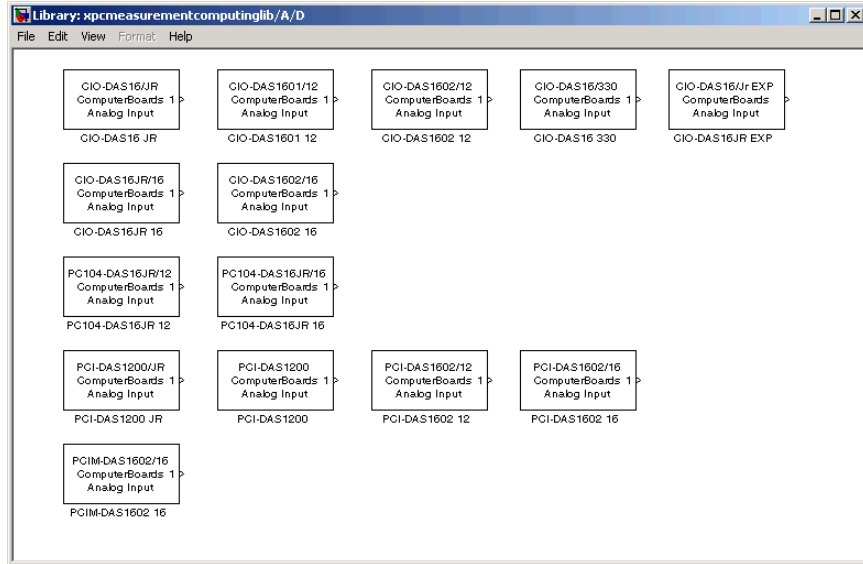
The manufacturer level opens.



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

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

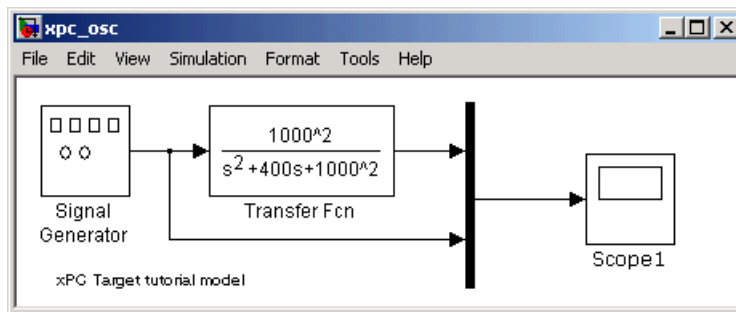
The window with the A/D driver blocks for Measurement Computing opens.



4 In the Simulink window, type

`xpc_osc`

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

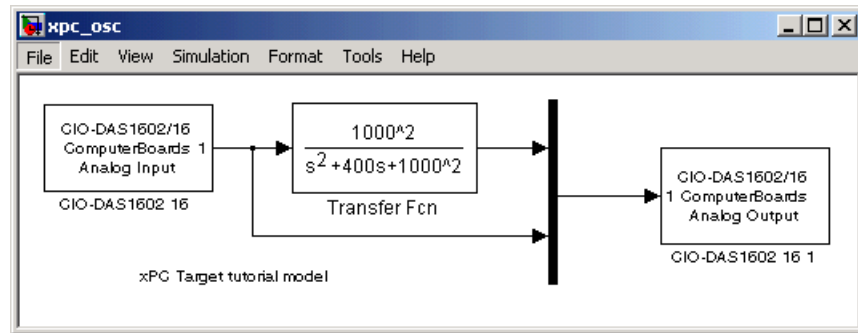


5 From the block library, click and drag the name of an A/D board to the Simulink block diagram. Likewise, click and drag the name of a D/A board to your model.

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

- 6 Remove the Signal Generator block and add the Analog Input block in its place. Remove the Scope block and add the Analog Output block in its place.

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



You cannot run this model unless you have the I/O board shown installed in your target PC. However, you can substitute the driver blocks for another I/O board that is installed in the target PC.

Your next task is to define the I/O block parameters. See “Defining I/O Block Parameters” on page 1-20.



## Adding I/O Blocks with the Simulink Library Browser

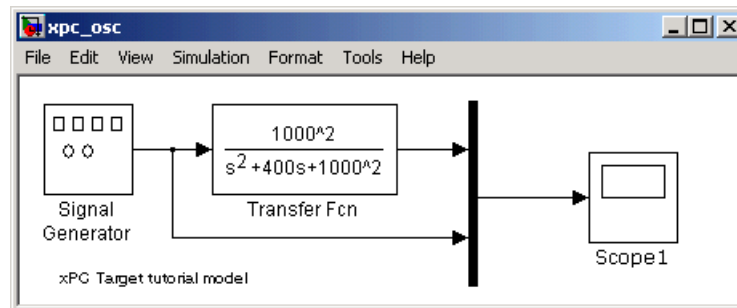
xPC Target includes a Simulink block library for I/O drivers. The highest hierarchical level in the library is grouped by I/O function. The second level is grouped by board manufacturer. The manufacturer groups within this second level contain the driver blocks for specific boards.

This procedure uses the Simulink model `xpc_osc.mdl` as an example of how to add and connect I/O blocks:

- 1 In the MATLAB window, type

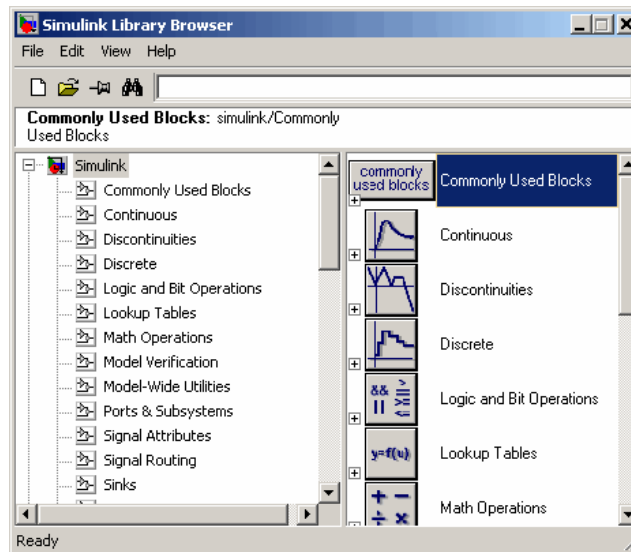
```
xpc_osc
```

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



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

The Simulink Library Browser window opens. Alternatively, you can open the Simulink Library Browser by typing `simulink` in the MATLAB Command Window.

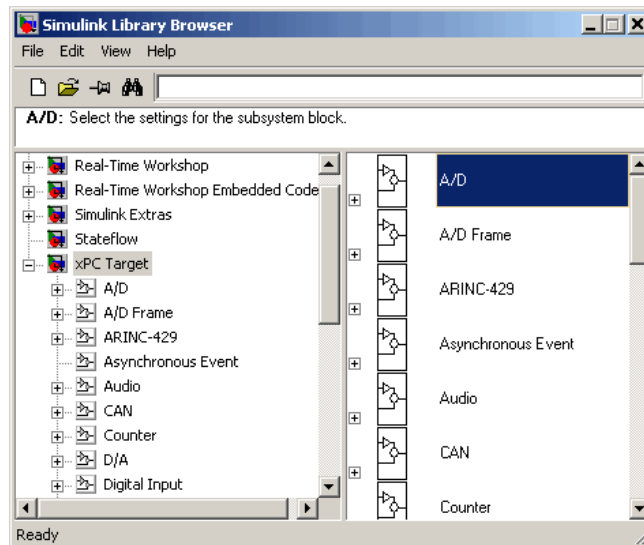


You can access the xPC Target I/O library by right-clicking **xPC Target** and then clicking **Open the xPC Target Library**.

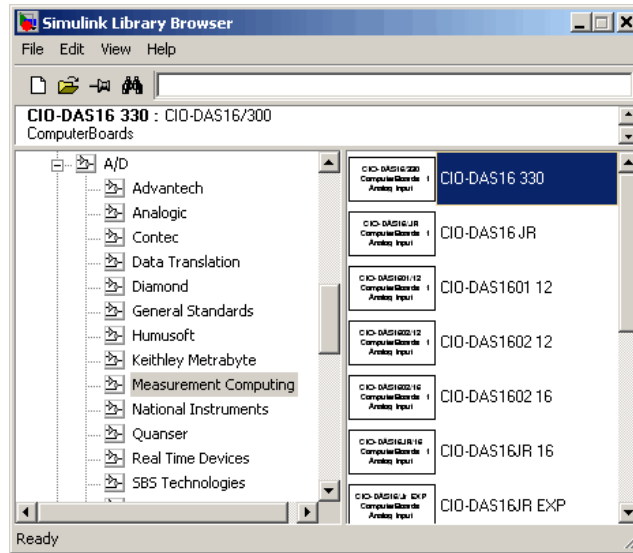
Alternatively, you can access driver blocks using the xPC Target I/O driver library. See “Adding I/O Blocks with the xPC Target Library” on page 1-11.

**3 Double-click xPC Target.**

A list of I/O functions opens.

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

A list with the A/D driver blocks for Measurement Computing opens.

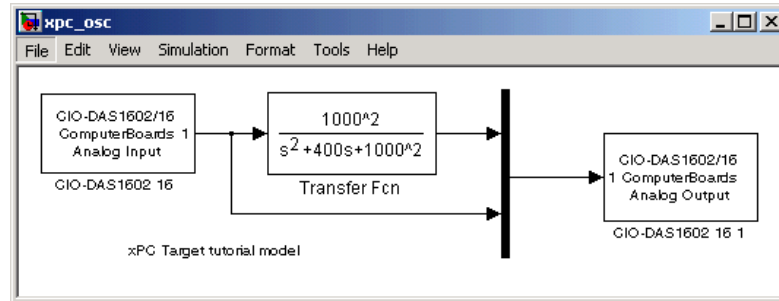


- From the block library, click and drag the name of an A/D board to the Simulink block diagram. Likewise, click and drag the name of a D/A board to your model.

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

- Remove the Signal Generator block and add the analog input block in its place. Remove the Scope block and add the analog output block in its place.

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



You cannot run this model unless you have the I/O board shown above installed in your target PC. However, you can substitute the driver blocks for another I/O board that is installed in the target PC.

Your next task is to define the I/O block parameters. See “Defining I/O Block Parameters” on page 1-20

## Defining I/O Block Parameters

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

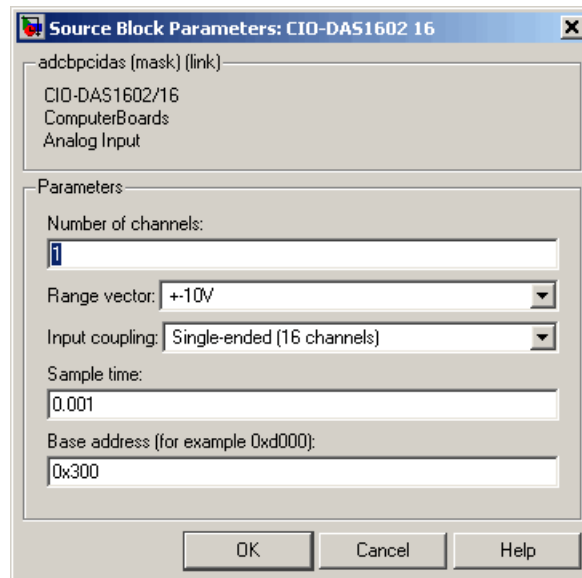
This procedure uses the Simulink model `xpc_osc.mdl` as an example, and assumes you have added an analog input and an analog output block to your model. To add an I/O block, see either “Adding I/O Blocks with the xPC Target Library” on page 1-11 or “Adding I/O Blocks with the Simulink Library Browser” on page 1-15

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

The dialog box for the A/D converter opens.

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

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

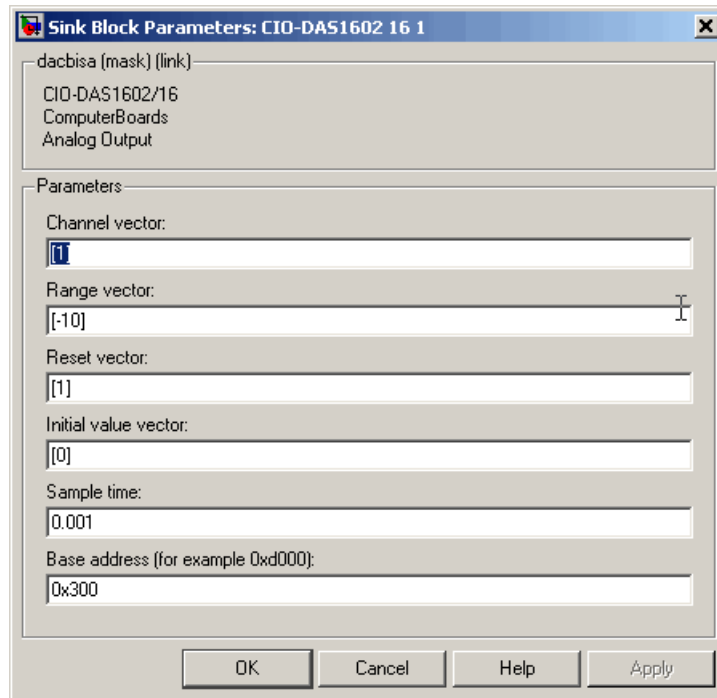


- 3** In the Simulink window, double-click the output block labeled **Analog Output**.

The dialog box for the D/A converter opens.

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

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



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

Your next task is to build and run the target application. See “xPC Target Application” in Getting Started with xPC Target.



# Serial Communications Support

---

xPC Target interfaces the target PC to serial devices using either the COM1 or COM2 port of the main board, through Quatech drivers, through Diamond Systems drivers, or through Commtech drivers. This chapter includes the following sections:

- |  |   |
|--|---|
| Introduction to Serial Drivers (p. 2-2)                    | Description of hardware connections and host/target PC communications.  |
| xPC Target RS-232 and 422/485 Drivers (Composite) (p. 2-5) | Description of composite xPC Target RS-232/422/485 drivers. Includes description of procedure to add an RS-232 driver block to your Simulink model. Also describes associated Simulink blocks.  |
| xPC Target RS-232 Drivers (Conventional) (p. 2-19)         | Description of conventional xPC Target RS-232 drivers. Includes description of procedures to add an RS-232 driver block to your Simulink model and create the message structures associated with those blocks. Also describes associated Simulink blocks and MATLAB message structures associated with the Simulink blocks. |
| Boards and Blocks — Alphabetical List (p. 2-53)            | Description of block parameters for serial board driver blocks  |

## Introduction to Serial Drivers

xPC Target supports RS-232 I/O communication with the following:

- Serial ports on the target PC
- Third-party Quatech PCI boards (<http://www.quatech.com>)
- Third-party Diamond Systems PC/104 boards (<http://www.diamondsystems.com>)

For the target PC serial ports, xPC Target can use these ports as the RS-232 I/O devices. You can initiate RS-232 communications with these ports and the accompanying xPC Target drivers.

xPC Target also supports the following:

- RS-232 — QSC-100 and ESC-100 PCI boards from Quatech
- RS-422, RS-485 — QSC-200/300 PCI boards and DSCP-200/300 dual channel PXI boards from Quatech, Fastcom: 422/2-PCI adapter from Commtech (<http://www.commtech-fastcom.com>)
- RS-232, RS-422, RS-485 — Emerald-MM and Emerald-MM-8 PC/104 boards from Diamond Systems. These boards provide 4 and 8 serial ports, respectively. These boards are jumper-configurable for the protocols (see the manufacturer documentation for details).

xPC Target provides a set of functionally similar drivers for these boards. See “RS-232/422/485 Simulink Block Reference” on page 2-13 for a description of the driver blocks that support the different protocols.

xPC Target supplies two types of drivers to support RS-232 I/O communication, conventional and composite:

- The composite drivers support RS-232 I/O for the target PC serial ports, the Quatech RS-232, RS-422, and RS-485 I/O devices, and the Diamond Systems RS-232 I/O devices. These drivers support communication in asynchronous binary mode. xPC Target uses Simulink blocks for the I/O drivers. The composite drivers provide a simple ASCII encode/decode for the send and receive RS-232, RS-422, and RS-485 blocks. This set of drivers has the descriptive name “composite” because the driver represents

each functional piece of the driver as a Simulink block. For more precise behavior, you can customize the RS-232 driver with these blocks.

- The conventional drivers support RS-232 I/O only for the target PC serial ports. These drivers support synchronous, asynchronous, and binary (asynchronous) communication mode. xPC Target uses a model for this RS-232 I/O that includes both Simulink blocks for the I/O drivers and MATLAB structures for sequencing messages and commands.

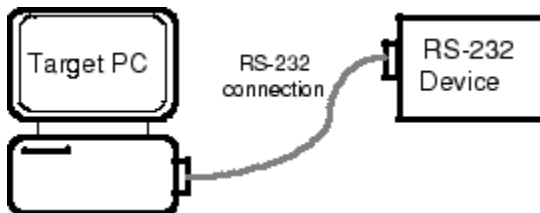
This section includes the following topics:

Hardware Connections for RS-232 (p. 2-3)	Connect the target PC to an RS-232 device.
Host and Target PC Communication (p. 2-4)	Consider limitations to using RS-232 for I/O on the target PC when using RS-232 communication between the host PC and target PC.

## Hardware Connections for RS-232

xPC Target supports serial communication with the COM1 and COM2 ports on the target PC.

Your target applications can use these RS-232 ports as I/O devices. Typically, the target PC is connected to an RS-232 device with a NULL modem cable. However, this depends on the DTE/DCE configuration of the devices, and you might not use a NULL modem cable.



### Host and Target PC Communication

If the host PC and target PC are connected using serial communication, one COM port on the target PC is dedicated for communication with the host PC. You cannot use this COM port in your block diagram as an I/O device.

For example, if the target PC uses COM1 for the communication with the host PC, COM1 cannot be used by your block diagram. If you try to use COM1 as an I/O device in your block diagram, an error message is displayed. The error message appears when you attempt to build and download the target application. In this example, you must use COM2 as an I/O device in your block diagram.

If you are using TCP/IP as your host PC to target PC communications protocol, then you can use any of the COM ports for RS-232 I/O.

---

**Note** When you use composite driver blocks, COM1 and COM3 often share interrupt line 4. Similarly, COM2 and COM4 often share interrupt line 3. If you use COM1 for host-target communication, you cannot also use COM1 or COM3 in a model. This is because the shared interrupt is caught in the xPC Target operating system. However, if COM3 uses an interrupt different from that for COM1, you can use COM3 in a model while using COM1 for host-target communications. If COM1 and COM3 share an interrupt line, you can use COM2 or COM4 as your RS-232 I/O port.

---

## xPC Target RS-232 and 422/485 Drivers (Composite)

This section describes the components that make up the RS-232 and RS-422/485 composite drivers, and how you can create a model using these drivers. These drivers perform RS-232 or RS-422/485 asynchronous communications.

xPC Target supports the target PC serial ports (main board), Quatech RS-232/422/485 devices, Diamond Systems RS-232 devices, and Commtech Fastcom: 422/2-PCI adapters with composite drivers. These drivers distribute the functionality of the device across several subsystems and blocks. For most RS-232/422/485 requirements, you can use these RS-232/422/485 drivers as they are implemented. However, if you need to customize the xPC Target RS-232/422/485 drivers, the composite nature of the target PC serial port, Quatech RS-232/422/485, Diamond Systems RS-232, and Commtech Fastcom drivers enables you to do so. See “RS-232/422/485 Internal Blocks and Subsystems” on page 2-18 for details.

Note the following characteristics of the Commtech Fastcom: 422/2-PCI adapter boards (<http://www.commtech-fastcom.com>):

- The Fastcom 422/2-PCI board has only two independent RS-422 channels.
- The Fastcom 422/2-PCI board can handle baud rates up to 1.5 megabaud.
- The Fastcom 422/2-PCI board hardware FIFO is fixed at 128 bytes for receive and transmit.

This section includes the following topics:

- “Adding RS-232 Blocks” on page 2-6
- “Building and Running the Target Application (Composite)” on page 2-12
- “RS-232/422/485 Simulink Block Reference” on page 2-13

---

**Note** Many of the blocks that support the RS-232 and RS-422/485 composite drivers are common across the main board, Quatech, and Diamond Systems boards. The descriptions for these blocks are applicable for all drivers, with specific board notes as appropriate.

---

### Adding RS-232 Blocks

You add RS-232 subsystem blocks to your Simulink model when you want to use the serial ports on the target PC, Quatech QSC-100 or ESC-100, or Diamond Systems Emerald-MM or Emerald-MM-8 serial device connected to the target PC, for I/O.

After you create a Simulink model, you can add xPC Target driver blocks and configure those blocks. The following procedure describes how to use the serial ports on the target PC for I/O with the composite drivers.

Before you start, decide what COM port combinations you want to use. The example has you configure the Baseboard Send/Receive block. To properly configure this block, you need to select serial port pairs. This parameter specifies the ports for which you are defining transmit and receive. You have a choice of the following:

- Com1 / none
- Com2 / none
- Com1 / Com3
- Com2 / Com4
- none / Com3
- none / Com4
- Custom

If you choose either the Com1 / Com3 or Com2 / Com4 pair, check that the port pair shares an interrupt. If the port pair does not share an interrupt, you cannot use the two ports as a pair.

Alternatively, you can define a Custom port pair. A Custom port pair is one that does not match the existing combinations of port pairs. When you select Custom, the dialog allows you to configure your own port pair. For example, you can set the IRQ and two addresses for the port pair. If one of the ports is not used, set that address to 0.

Normally, the ports are set to the following:

COM1 — 0x3F8, IRQ 4

COM2 — 0x2F8, IRQ 3

COM3 — 0x3E8 (if present), IRQ 4

COM4 — 0x2E8 (if present), IRQ 3

A Custom port pair is one where one or both ports of the pair are set to addresses other than these conventions, or one for which you want to assign a different IRQ value. Some hardware allows you to set the IRQ numbers independently.

If you choose the port pairs Com1/Com3 or Com2/Com4, you need to include one Send/Receive subsystem block in the model. If you choose to use COM1 and COM2, or COM1 and a custom port pair, you need to include two Send/Receive blocks in the model.

The following example shows two models, one that uses a standard Com1/Com3 port pair, and one that uses custom port pairs:

**1** In the MATLAB Command Window, type

```
xpcLib
```

The xPC Target driver block library opens.

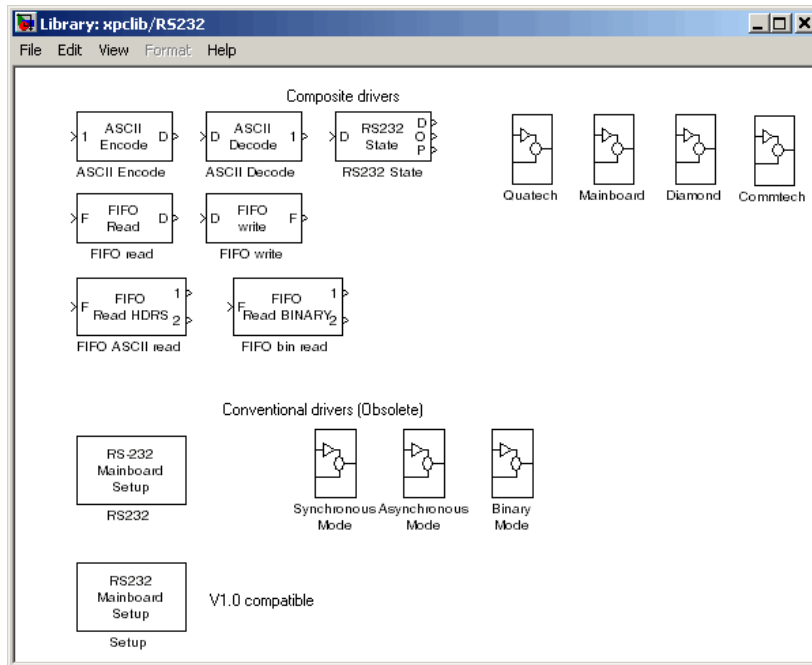
2 Double-click the RS-232 group block.

A window with blocks for RS-232 drivers opens.

---

**Note** This library contains two sections, composite and conventional.

---



Alternatively, you can access the xPC Target block library from the Simulink Library Browser. In the Simulink window, and from the **View** menu, click **Show Library Browser**. In the left pane, double-click **xPC Target**, and then click **RS232**.

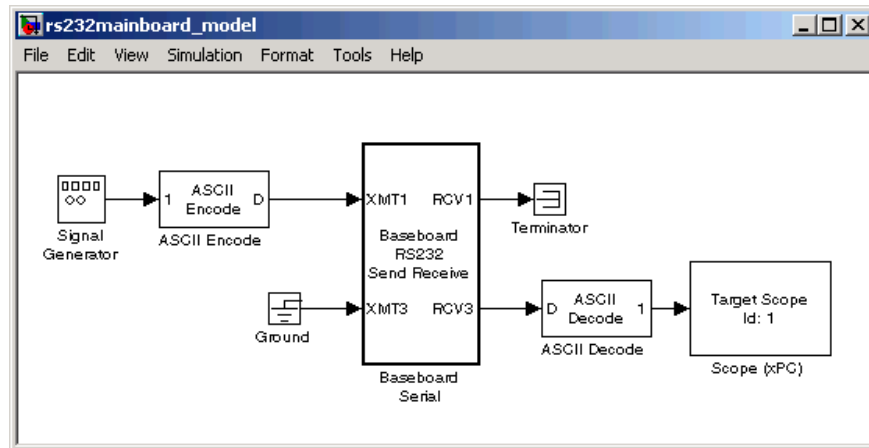
3 Drag and drop an ASCII Encode block to your Simulink model. This block encodes input for the RS-232 Send Receive block.

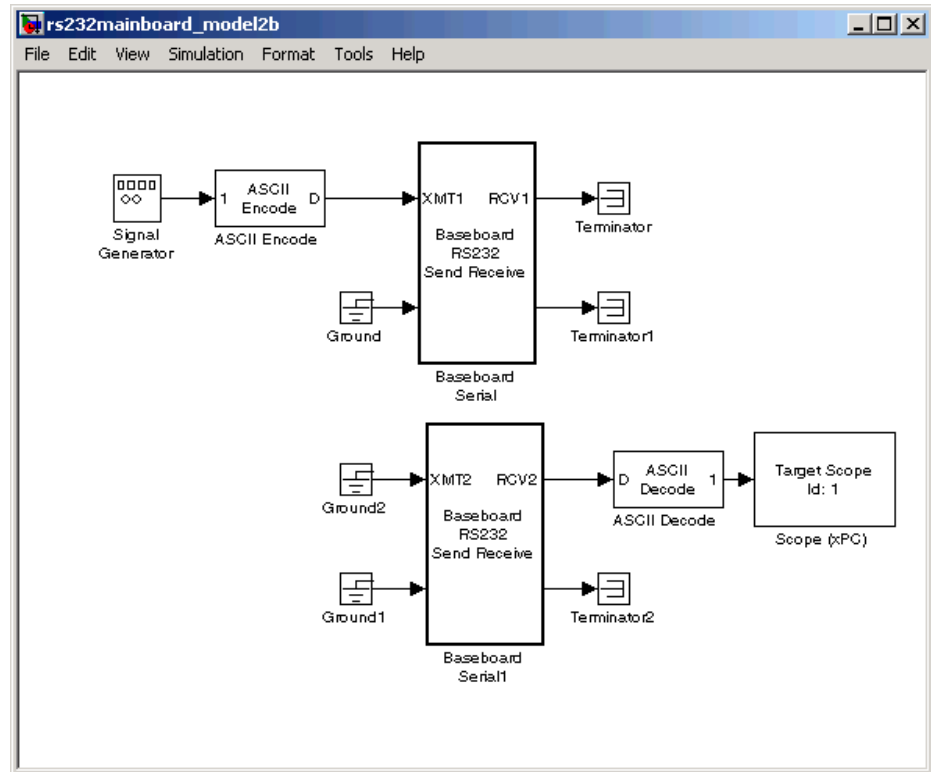
4 Configure this block.



- 5** Drag and drop an ASCII Decode block to your Simulink model. This block decodes output from the RS-232 Send Receive block.
- 6** Configure this block.
- 7** Double-click the Mainboard group block.
- 8** Depending on your port pair configuration, drag and drop one or two Baseboard RS-232 Send/Receive blocks to your Simulink model.
- 9** Double-click the Baseboard RS-232 Send/Receive block.
- 10** Configure this block. Pay particular attention to the **Parameter** group Board Setup entry.
- 11** Add a Signal Generator and Target Scope block.
- 12** From the Simulink Library Browser, select **Sinks**. Depending on your configuration, drag and drop one or more Terminator blocks. Connect this block to the unused RCV1 port to suppress unused port messages.
- 13** From the Simulink Library Browser, select **Sources**. Depending on your configuration, drag and drop the Ground block. Connect this block to the unused XMT3 port to suppress unused port messages.

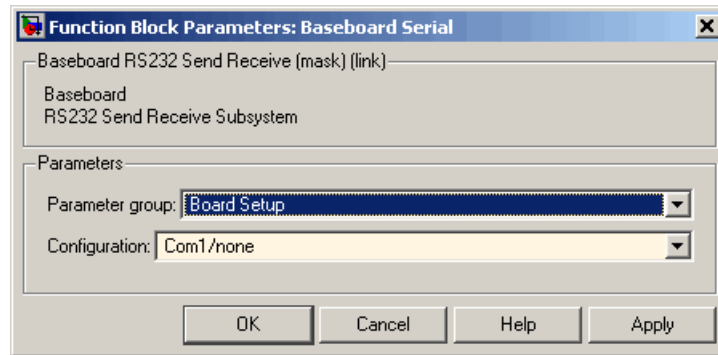
Your model should look similar to one of the following figures. The first figure shows a single-block model. This model uses the Com1 / Com3 port pair. The second figure shows a two-block model. This model uses two sets of Custom port pairs.





- 14** Double-click a Baseboard RS232 Send Receive block. Enter values to configure the port(s) on the target PC for this board.

For example, if the target PC is connected to COM1, your Send Receive block dialog box should look similar to the following figure. Note, this is a dynamic dialog box that changes depending on the **Parameter group** selection.



For more information on entering the block parameters, see RS-232/RS-422/RS-485 Send/Receive.

**15** Click **OK**. The Send Receive block dialog box dialog box closes.

Your next task is to build and run the target application.

## Building and Running the Target Application (Composite)

xPC Target and Real-Time Workshop® create C code from your Simulink model. You can then use a C compiler to create executable code that runs on the target PC.

After you have added the RS-232 blocks for the main board to your Simulink model, you can build your target application.

---

**Note** You cannot use a serial port to communicate between the host PC and target PC with this example. You can only use COM1 if it is not already in use for host-target communications. Additionally, if COM1 and COM3 share an interrupt, you cannot use COM3 if COM1 is already in use for host-target communications.

---

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

**2** In the MATLAB Command Window, type

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

## **RS-232/422/485 Simulink Block Reference**

xPC Target supports RS-232/422/485 communication with driver blocks in your Simulink model.

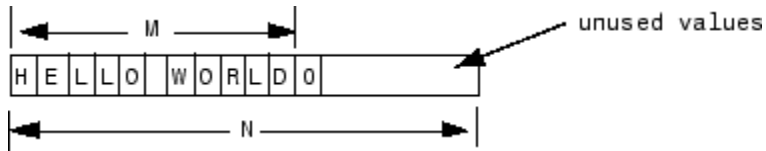
This section includes the following topics:

- “Signal Data Types” on page 2-14 — Describes signal data types that composite drivers support.
- ASCII Encode/Decode — (Generic) Describes encoder and decoder blocks. Encoders convert input signals for the send/receive subsystem to ASCII strings. ASCII decoders parse the string from the Send/Receive subsystem.
- FIFO Read/Write — (Generic) Describes FIFO read and write blocks.
- RS232 State — (Generic) Monitors the hardware error state information that is present in the output vector from all blocks.
- RS-232/RS-422/RS-485 Send/Receive — Provides blocks for sending and receiving.
- Modem Control — Controls the state of either or both of the RTS and DTR output lines.
- Modem Status — Reads the states of the four input modem control lines.
- “RS-232/422/485 Internal Blocks and Subsystems” on page 2-18 — Provides blocks to customize the RS-232/422/485 drivers.

### Signal Data Types

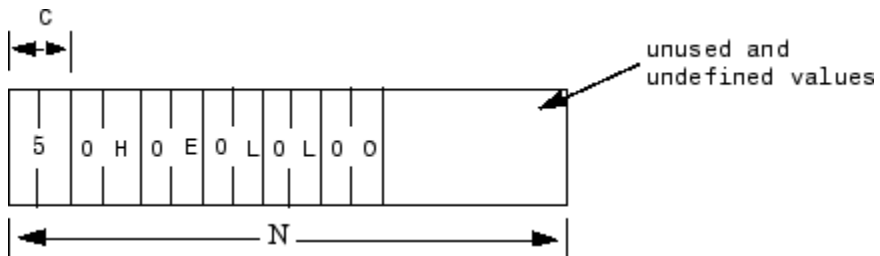
Signals between blocks in composite drivers can be one of several basic data types, 8-bit and 16- or 32-bit. Both of these types are structures.

8-bit data types are NULL-terminated strings that are represented as Simulink vectors. The width is the maximum number of characters that can be stored. In the following figure, M is the actual set of stored characters and N is the maximum number of characters that can be stored.



This string has 11 characters terminated with a NULL byte (0). This data type cannot contain a NULL byte as part of the real data.

16- and 32-bit data types use the first element of the vector as a count of the valid data. In the following figure of a 16-bit data type, C is the count of the valid data, N is the width of the vector.



These serial blocks interpret each entry in the vector as a single character. The low-level hardware Send block writes the low-order byte of each entry to the UART. The 16- and 32-bit data types allow the embedding of any 8-bit data value, including 0.

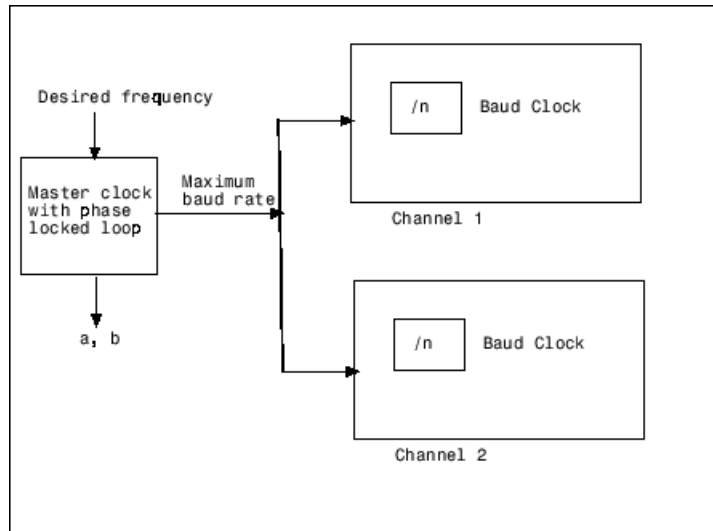
The 8-bit data type is most useful with the ASCII Encode and Decode blocks. The 16- and 32-bit data types are most useful for binary data streams.

## Defining the Commtech Fastcom Baud Rate

The Commtech Fastcom 422/2-PCI board can handle baud rates up to 1.5 megabaud. To configure a baud rate for the board, you need to set the following parameters:

- **Clock Bits** in the Fastcom 422/2-PCI Send Receive block with the **Parameter group** parameter set to **Board Setup**
- **Baud Divisor** in the Fastcom 422/2-PCI Send Receive block with the **Parameter group** parameter set to **Basic Setup**

The Fastcom 422/2-PCI board has two serial channels, each of which has an independent counter (baud clock). A master clock generator, which has a phase locked loop, controls the master clock for both serial channels. The master clock generates a maximum baud rate for both channels. The block determines the actual baud rate of a channel by dividing the maximum baud rate from the master clock by the baud rate divisor ( $n$ ).



Fastcom 422-2 PCI Board

To correctly set the block parameters for this board, choose a maximum baud rate, as follows. This procedure assumes that both channels require different

baud rates. Determine a common base clock that can be divided to produce the required baud rates for both channels, as follows:

- 1** In the MATLAB Command Window, type a command like the following. With a desired frequency (for example,  $1.5e6$ ) as the desired input, the `fc422mexcalcbits` utility calculates parameter values that you can use to configure the board rate for your board.

```
[a b df ] = fc422mexcalcbits(1.5e6)
```

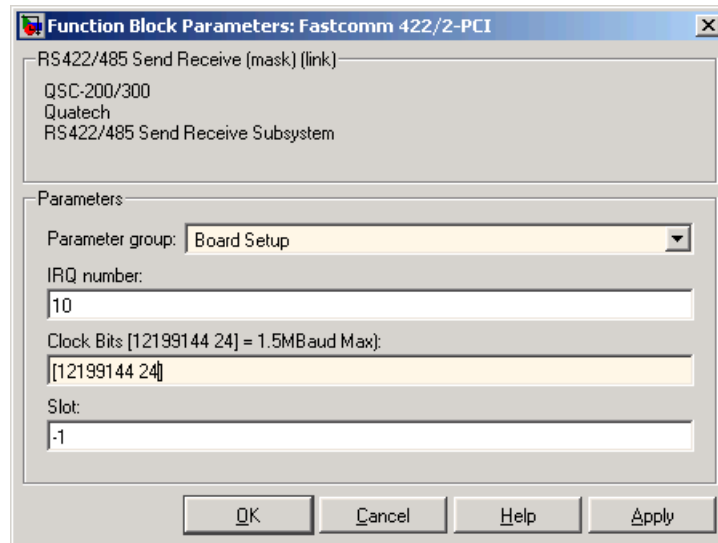
This command returns three values.

```
a = 12199144  
b = 24  
df = 1500000
```

- 2** Examine the `df` value. The `df` value is the actual frequency the board will be able to attain compared to your desired frequency. If the actual attainable frequency is not accurate enough, you might want to try another frequency. In this example, the board can match the desired frequency of  $1500000$  ( $1.5e6$ ).



- 3** Enter the first two values in the **Clock Bits** parameter of the Fastcom 422/2-PCI Send Receive block with the **Parameter group** parameter set to Board Setup.



After you define a maximum baud rate, you can set a unique baud rate for each channel by choosing a different baud rate divisor for each channel. For example, you can have Channel 1 have a baud rate of 750000 (1500000/2) and Channel 2 have a baud rate of 1500000 (1500000/1). To set Channel 1 to have a baud rate of 750000, with the Fastcom 422/2-PCI Send Receive block with the **Parameter group** parameter set to Basic Setup, set

- **Port to modify** to 1
- **Baud Divisor** to 2

---

**Note** For very slow baud rates (less than 30000), you must use the **Baud Divisor** parameter to achieve the desired baud rate.

---

### **RS-232/422/485 Internal Blocks and Subsystems**

This section describes the internal blocks of the RS-232/422/485 boards. Typically, the parameters in these blocks are controlled from the mask parameters dialog for the send/receive subsystem in which they are used.

You might need to access these blocks if you need to modify the RS-232 Quatech subsystems for your use.

---

**Note** Otherwise, do not use these blocks directly.

---

This section includes the following topics:

- RS-232/422/485 Setup (Composite) — Sets up the interface characteristics for the board.
- RS-232/422/485 Read Hardware FIFO (Composite) — Reads characters from the hardware FIFO in the UART.
- RS-232/422/485 Write Hardware FIFO (Composite) — Writes the data from the input port to the hardware FIFO in the UART for this port.
- RS-232/422/485 Read Int(errupt) Status (Composite) — Reads the interrupt status for the boards in the system.
- RS-232/422/485 Enable TX Interrupt (Composite) — Enables the transmitter buffer empty interrupt when data is present in the software FIFO.
- RS-232/422/485 Filter Interrupt Reason (Composite) — Filters the output of the Read Int(errupt) Status block.
- RS-232/422/485 Board Setup (Commtech) and Interrupt Check (Quatech) (Composite) — Checks for instances where the hardware IRQ differs from the software for which it is listening.

## xPC Target RS-232 Drivers (Conventional)

This section describes the components that make up the RS-232 conventional drivers, and how you can create a model using these drivers. This section includes the following topics:

- “Simulink Blocks for RS-232 I/O (Conventional)” on page 2-19 — Add setup, send, send/receive, and receive blocks to your Simulink model.
- “MATLAB Message Structures for RS-232 I/O (Conventional)” on page 2-20 — Create message structures to sequence instructions to and from the RS-232 device.
- “RS-232 Synchronous Mode (Conventional)” on page 2-21 — Add synchronous driver blocks to have the device wait for a response before continuing with other computations.
- “RS-232 Asynchronous Mode (Conventional)” on page 2-31 — Add asynchronous driver blocks if the device does not have to wait for a response before continuing with other computations.
- “RS-232 Simulink Block Reference (Conventional)” on page 2-44 — Description of the RS-232 blocks for the conventional drivers.
- “RS-232 MATLAB Structure Reference (Conventional)” on page 2-44 — Description of the RS-232 MATLAB structure for messages.
- “RS-232 Binary Mode (Conventional)” on page 2-48 — Add binary driver blocks to transfer raw data.

### Simulink Blocks for RS-232 I/O (Conventional)

To support the use of RS-232, the xPC Target I/O library includes a set of RS-232 driver blocks. These driver blocks can be added to your Simulink model to provide inputs and outputs using one or more of the RS-232 ports.

- **RS-232 Setup** — One setup block is needed for each RS-232 port you use in your model. The setup block does not have any inputs or outputs, but sends the initialization and termination messages.
- **RS-232 Send/Receive (Synchronous Mode)** — Send/Receive blocks have inputs and outputs from your Simulink model, and wait for responses to messages sent and received.

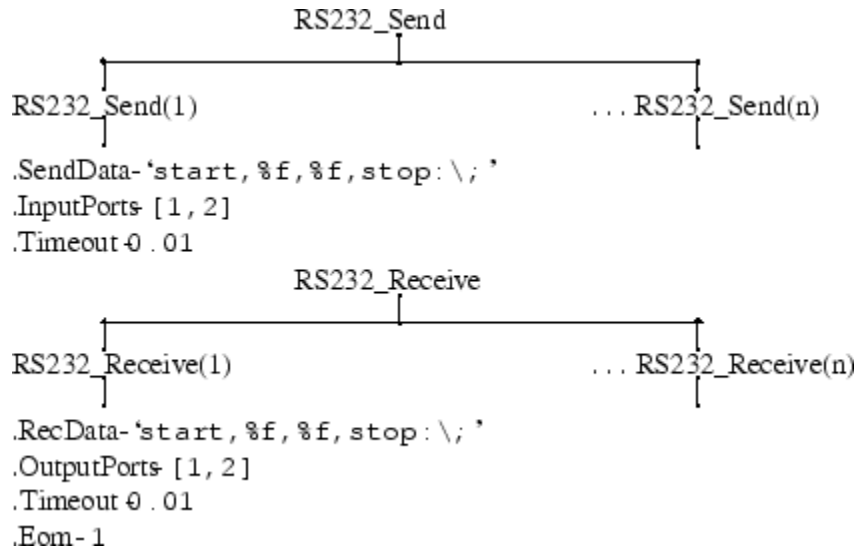
- **RS-232 Send (Asynchronous Mode)** — Send blocks have inputs from your Simulink model, and wait for responses to messages sent.
- **RS-232 Receive (Asynchronous Mode)** — Receive blocks have output from your Simulink model, and wait for responses to messages received.

### **MATLAB Message Structures for RS-232 I/O (Conventional)**

Communication is through a series of messages passed back and forth between the target PC and the RS-232 device. To accomplish this, the messages sent to the RS-232 device must be in a format that the device understands. Likewise, the target PC must know how to interpret the data returned from the RS-232 device.

xPC Target uses MATLAB structures to create messages and map the input and output ports on the RS-232 driver blocks to the data written and read from the RS-232 devices. The RS-232 Setup block sends the messages in the initialization structure after downloading the target application. The RS-232 Send/Receive, RS-232 Send, and RS-232 Receive blocks repeat the sending of the messages in the send/receive, send, and receive structures during each sample interval. When the target application stops running, the RS-232 Setup block sends the messages in the termination structure.

Below is an example of the send and receive message structure for asynchronous communication. In this example, an external RS-232 device requires a string with two floating-point numbers. The numbers are entered from the Simulink model to the first and second input ports of the RS-232 Send driver block. The RS-232 device sends back two floating-point numbers that are passed to the outputs of the RS-232 Receive driver block.



For more information on this example, see “Creating RS-232 Message Structures (Asynchronous)” on page 2-39.

## RS-232 Synchronous Mode (Conventional)

Use synchronous mode when you need to receive a response before continuing with other computations. In synchronous mode, data is sent to an external device and the driver block waits for a response. In other words, the I/O driver *blocks* or stops execution of the target application until an answer is received from the external device or it reaches a time-out. This section includes the following topics:

- “Notes for RS-232 Synchronous Mode” on page 2-22 — Overview of RS-232 communication with xPC Target blocks.
- “Adding RS-232 Driver Blocks (Synchronous)” on page 2-22 — Add the setup, send, and receive blocks you need to your Simulink model for RS-232 communication.
- “Creating RS-232 Message Structures (Synchronous)” on page 2-28 — Create the initialize, send/receive, and termination message structures you need in the MATLAB workspace.

### Notes for RS-232 Synchronous Mode

For the example in this section, assume an external device (RS-232 device) includes a D/A conversion module with four independent channels and an output voltage range of -10 to 10 volts. Also assume that the external device outputs a new voltage if it receives a serial string with a value to identify the D/A channel and the voltage value.

Use a Constant block as an input to the Send/Receive block to select the D/A channel, and a Signal Generator block as a source for voltage values. Also, set up the message structures to receive a confirmation message from the external module after the target PC sends a message string to the device.

In synchronous mode, the data is sent to the external device and the block waits until a response (for example, data) is received from the device before the execution of the block is considered to be complete. In other words, the I/O driver *blocks* until an answer is received from the external device or it reaches a time-out.

When it is necessary to receive a response before continuing with other computations, synchronous mode is used, which implies that the Send & Receive block is placed in your model. This block includes both input and output lines.

### Adding RS-232 Driver Blocks (Synchronous)

You add RS-232 driver blocks to your Simulink model when you want to use the serial ports on the target PC for I/O.

After you create a Simulink model, you can add xPC Target driver blocks and define the initialization, send/receive, and termination message structures:

- 1 In the MATLAB command window, type

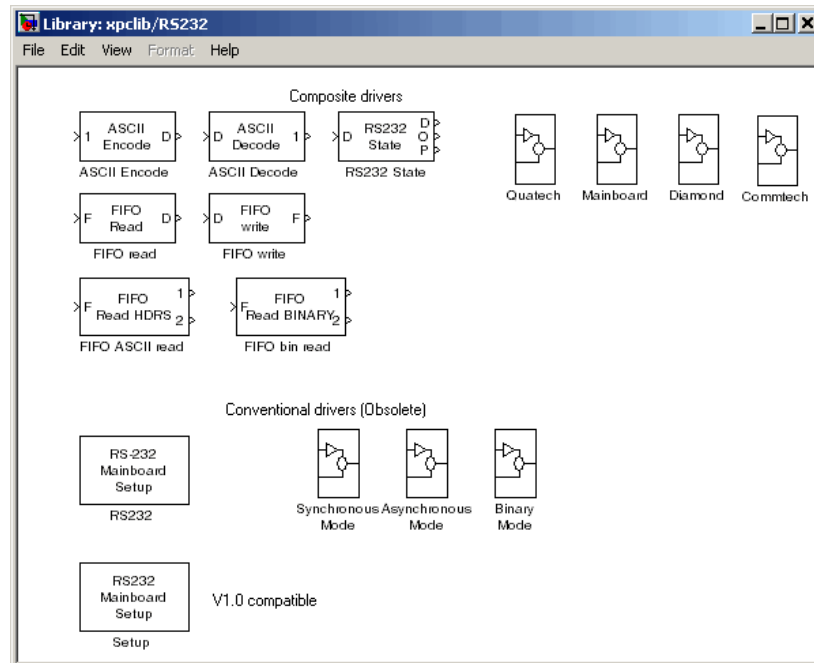
```
xpcLib
```

The xPC Target driver block library opens.

- 2 Double-click the RS-232 group block.

A window with blocks for RS-232 drivers opens.

**Note** This library contains two main sections, Composite drivers and Conventional drivers. Refer to the Conventional drivers section, where there are two setup blocks. The second block is included for compatibility with xPC Target Version 1.0.

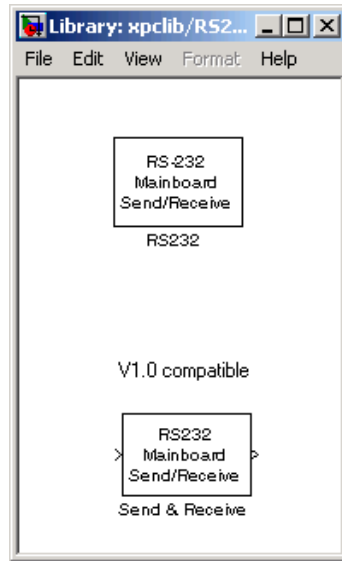


Alternatively, you can access the xPC Target block library from the Simulink Library Browser. In the Simulink window, and from the **View** menu, click **Show Library Browser**. In the left pane, double-click **xPC Target**, and then click **RS-232**.

- 3** From the Conventional drivers area, drag and drop an RS-232 Setup block to your Simulink model.
- 4** In the Library window, double-click the RS-232 Synchronous mode group block. The library window with blocks for RS-232 synchronous communication opens.

**Note** This library contains two Setup and Receive blocks. The second block is included for compatibility with xPC Target Version 1.0.

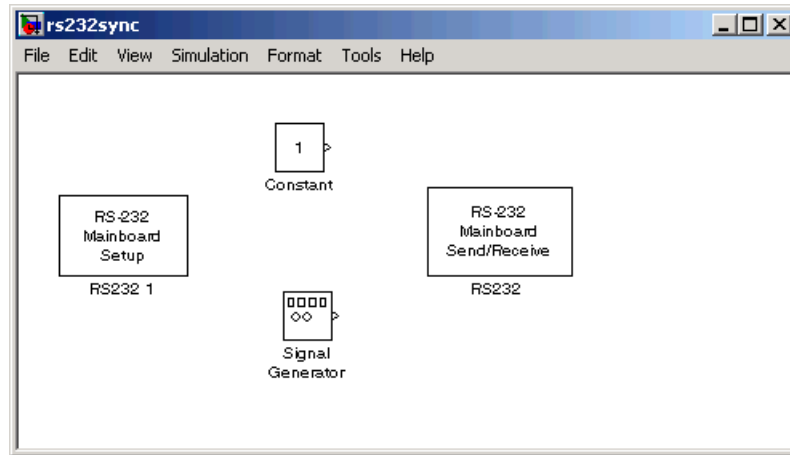
---



- 5 Drag and drop an RS-232 Send/Receive block to your Simulink model.
- 6 Add a Signal Generator and a Constant block.



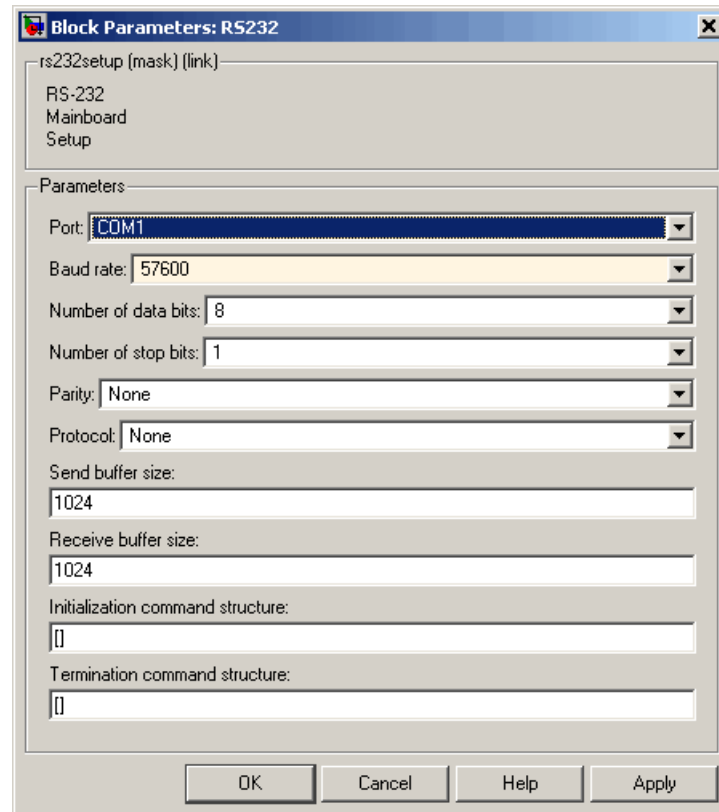
Your model should look similar to the figure shown below. Note that inputs on the RS-232 Send/Receive block are not defined or visible. The inputs are defined in a MATLAB message structure, and visible only after you load that structure into the MATLAB workspace and update your Simulink model.



- 7 Double-click the RS-232 Setup block. Enter values to configure the COM1 port on the target PC.

For example, if the target PC is connected to COM1, and serial communication is set to 5760 baud, 8 data bits, and 1 stop bit, your Block Parameter dialog box should look similar to the figure shown below.

**Note** If you are not using an initialization or termination structure, in the **Initialization Struct** and **Termination Struct** boxes, enter the empty matrix [ ].



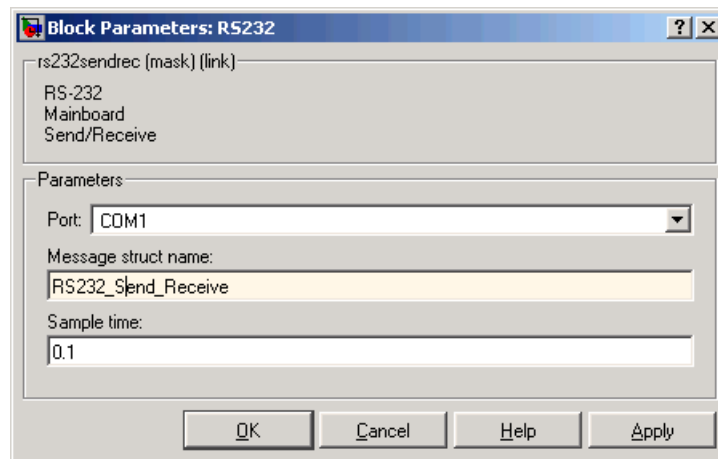
For more information on entering the block parameters, see RS-232 Mainboard Setup (Conventional). For the procedure to create the initialization and termination structures, see “RS-232 MATLAB Structure Reference (Conventional)” on page 2-44.

- 8 Click **OK**. The Block Parameters dialog box closes.

- 9 Double-click the RS-232 Send/Receive block. The Block Parameters dialog box opens.
- 10 From the **Port** list, select either COM1 or COM2. For this example, select COM1. In the **Message struct name** box enter the name for the MATLAB structure this block uses to send messages to the COM1 port. The name of the message structure is not the name of the M-file, but the name of the structure created with the M-file.

In the **Sample Time** box, enter the sample time or a multiple of the sample time you entered in the Receive block.

Your Block Parameter dialog box should look similar to the figure shown below.



For information on entering the block parameters, see RS-232 Mainboard Send/Receive (Synchronous) (Conventional). For the procedure to create the send/receive structure, see “RS-232 MATLAB Structure Reference (Conventional)” on page 2-44.

- 11 Click **OK**. The Block Parameters dialog box closes.

Your next task is to create the MATLAB message structures that the RS-232 driver blocks use to sequence commands to the RS-232 device. See “Creating RS-232 Message Structures (Synchronous)” on page 2-28.

## Creating RS-232 Message Structures (Synchronous)

RS-232 drivers use MATLAB structures to send and receive messages and map the input and output ports on the RS-232 driver blocks to the data written and read from the RS-232 devices.

After you add an RS-232 Setup and RS-232 Send/Receive block to your Simulink model, you can create the message structures to communicate with the RS-232 devices. You need to create and load these structures into the MATLAB workspace before you build your target application. The easiest way to create these structures is using an M-file and loading that M-file into the MATLAB workspace.

- 1** In the MATLAB Command Window, and from the **File** menu, point to **New**, and then click **M-file**.

A MATLAB text editor window opens.

- 2** Enter the initialization, send/receive, and termination messages. Each message is an element in a MATLAB structure array. For information and examples of this structure, see “RS-232 MATLAB Structure Reference (Conventional)” on page 2-44.

For example, assume that you have an external RS-232 device with a D/A module that requires a string in the format 'identifier, channel, value;\n'. identifier is any string. channel is an integer value between 1 and 2, defining which D/A channel to update. value is a floating-point value indicating the new voltage for the D/A output.

Additionally, when the external device receives a legal string, it accepts the string as an input message and returns the message 'noerror;\n'. This message is provided as a confirmation. As an example, you can type the following.

---

**Note** Field names in the structures are case sensitive.

---

```
RS232_Send_Receive(1).SendData = 'da_1234,%d,%f,;\n';  
RS232_Send_Receive(1).InputPorts = [1 2];  
RS232_Send_Receive(1).RecData = 'noerror\n';  
RS232_Send_Receive(1).Timeout = 0.01;  
RS232_Send_Receive(1).EOM = 1;
```

- 3** From the **File** menu, click **Save As**. In the Save as file dialog box, enter the name of the M-file script. For example, enter

```
RS232Sync_Messages.m
```

- 4** Close the text editing window.
- 5** In the MATLAB Command Window, type the name of the M-file you created with the RS-232 structures. For example, type

```
RS232Sync_Messages
```

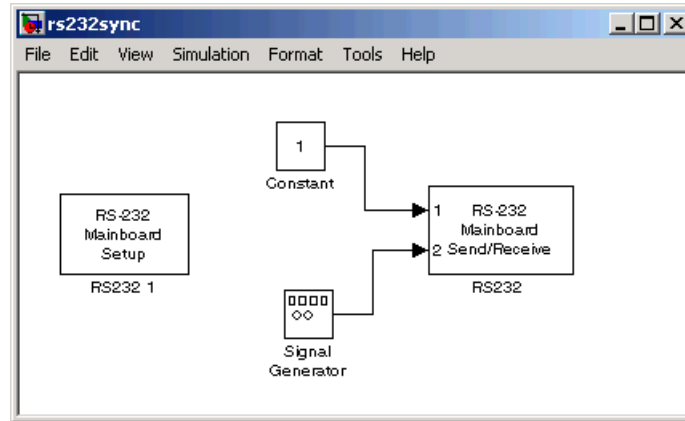
MATLAB loads and runs the M-file to create the message structures in the MATLAB workspace needed by the RS-232 driver blocks.

- 6** Open your Simulink model, or press **Ctrl+D**.

Simulink updates the RS-232 driver blocks with the information from the structures. For example, Simulink adds inputs and outputs defined in the structures to the blocks.

- 7** Connect the input and output ports on the RS-232 driver blocks to other blocks in your Simulink model.

Your model should look similar to the figure shown below.



- 8 Set the `PreLoadFcn` for your Simulink model to load the message structures when you open your model. For example, if you saved the message structures in the M-file `RS232Sync_messages`, type

```
set_param(gcs, 'PreLoadFcn', 'RS232Sync_messages.m')
```

---

**Note** If you do not manually load the message structures before opening your Simulink model, or have the message structures automatically loaded with the model, the port connections to the RS-232 driver break.

---

Your next task is to build and run the target application. However, the example above only illustrates how to set up the dialog entries when using the Send & Receive block. Without an external RS-232 device to receive the messages and return a reply 'no error\n', this model cannot run successfully on your target PC. It will *block* and wait for a reply each time the application sends a message.

## RS-232 Asynchronous Mode (Conventional)

Use asynchronous mode when you do not need a response before continuing with other computations. You can achieve faster sample rates with asynchronous mode because neither the Send nor Receive block waits for a reply. As a result, the asynchronous mode blocks do not *block* as do the synchronous mode blocks. The application updates the received outputs only when the entire package of data is received from the external device. This section includes the following topics:

Notes for RS-232 Asynchronous Mode (p. 2-31)	Overview of RS-232 communications with xPC Target blocks.
Adding RS-232 Driver Blocks (Asynchronous) (p. 2-32)	Add the setup, send, and receive blocks you need to your Simulink model for RS-232 communication.
Creating RS-232 Message Structures (Asynchronous) (p. 2-39)	Create the initialize, send/receive, and termination message structures you need in the MATLAB workspace.
Building and Running the Target Application (Asynchronous) (p. 2-42)	Run a real-time application with RS-232 communication.

### Notes for RS-232 Asynchronous Mode

For the example in this section, two asynchronous mode blocks illustrate how you can test RS-232 I/O on the target PC in a simple loop-back test. This simple but effective test lets you check that the RS-232 Send and RS-232 Receive blocks work correctly with your system using minimal hardware.

In this loop-back test, you use the COM1 port for sending signals and the COM2 port for receiving signals. A NULL modem serial cable connects COM1 to COM2 so that any messages sent from the target PC through COM1 are received by COM2 on the same target PC.

Use a Sine Wave block as an input to an RS-232 Send block that you connect to the COM1 port. Connect the COM2 port to an RS-232 Receive block. The signal received from this block is then passed through a Gain block of -1.

In the asynchronous mode, data is sent without waiting for response data to be received. The Send block completes execution immediately upon

completing the Send transfer. The Receive block completes execution upon completing the Receive transfer or when no more data is ready to be retrieved.

For sending data in asynchronous mode, use the RS-232 Send block. This block only has input lines for the data to be sent. For receiving data, you must use the Receive block. This block only has output lines for the data to be received. Outputs are updated only when the entire package of data is received from the external device.

### **Adding RS-232 Driver Blocks (Asynchronous)**

You add RS-232 driver blocks to your Simulink model when you want to use the serial ports on the target PC for I/O.

After you create a Simulink model, you can add xPC Target driver blocks and define the initialization, send, receive, and termination message structures:

**1** In the MATLAB Command Window, type

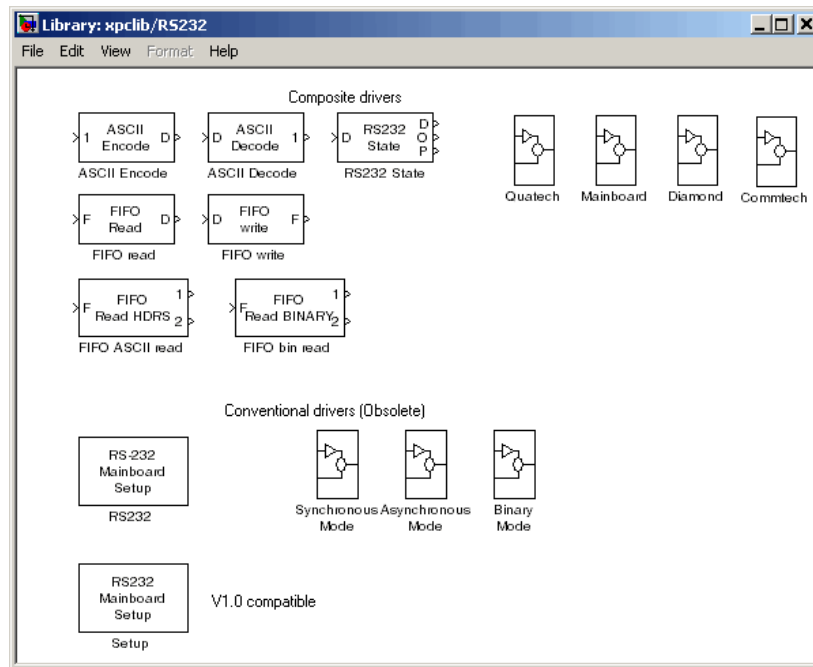
```
xpcLib
```

The xPC Target driver block library opens.



2 Double-click the RS-232 group block.

A window with blocks for RS-232 drivers opens.

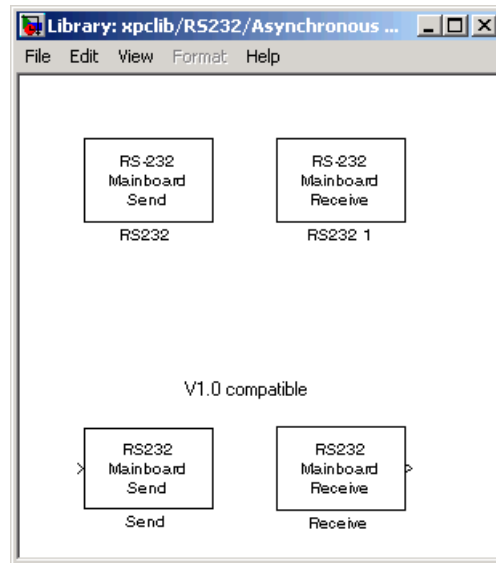


Alternatively, you can access the xPC Target block library from the Simulink Library Browser. In the Simulink window, and from the **View** menu, click **Show Library Browser**. In the left pane, double-click **xPC Target**, and then click **RS-232**.

3 Drag and drop two RS-232 Setup blocks to your Simulink model.

4 In the **Library** window, double-click the RS-232 Asynchronous mode group block. The library window containing blocks for RS-232 Synchronous communication opens.

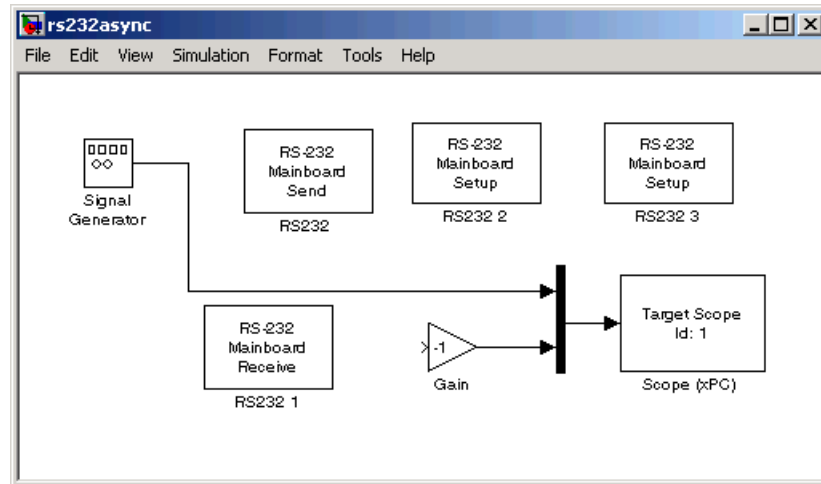
**Note** This library contains two send and two receive blocks. The second block is included for compatibility with xPC Target Version 1.0.



Alternatively, you can access the xPC Target block library from the Simulink Library Browser. In the Simulink window, and from the **View** menu, click **Show Library Browser**. In the left pane, double-click **xPC Target**, and then click **RS-232**.

- 5 Drag and drop the RS-232 Send and RS-232 Receive blocks into your Simulink model.
- 6 Add a Signal Generator, Gain, and xPC Target Scope block.

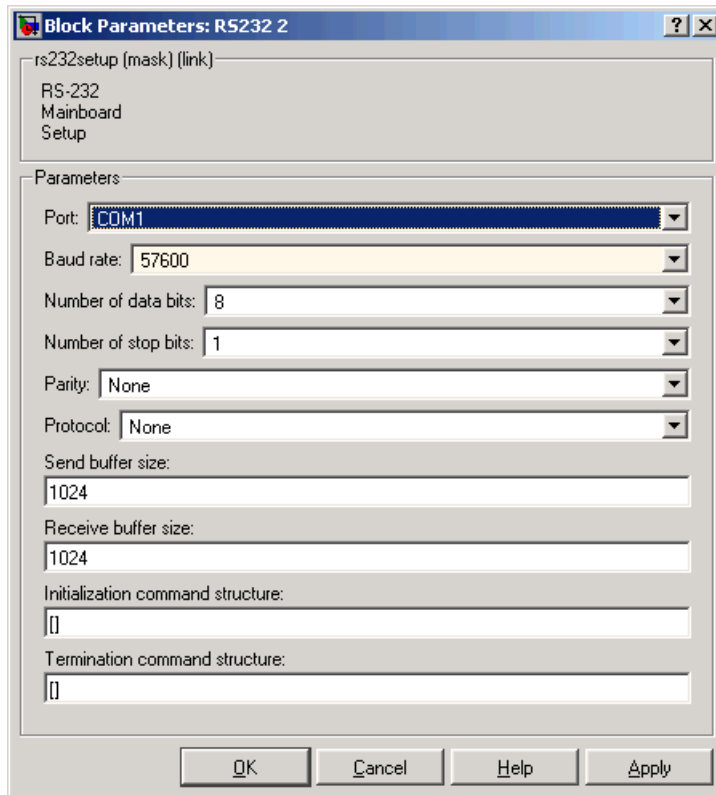
Your model should look similar to the figure below. Note that you cannot connect to the inputs on the RS-232 Send block and the outputs on the RS-232 Receive block, because they are not defined or visible. The inputs and outputs are defined in a MATLAB message structure, and visible only after you load that structure into the MATLAB workspace and update your Simulink model.



- 7 Double-click the first RS-232 Setup block. Enter values to configure the COM1 port on the target PC.

For example, if the COM1 and COM2 ports of the target are connected with a RS-232 NULL modem cable, and you set serial communication to 57600 baud, 8 data bits, and 1 stop bit. Your Block Parameters dialog box should look similar to the figure shown below.

**Note** If you are not using an initialization or termination structure, in the **Initialization Struct** and **Termination Struct** boxes, enter the empty matrix [ ].

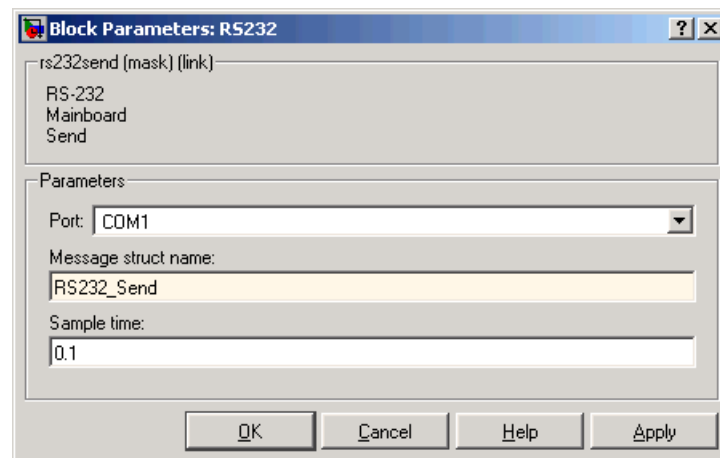


For more information on entering the block parameters, see RS-232 Mainboard Setup (Conventional). For the procedure to create the initialization and termination structures, see “RS-232 MATLAB Structure Reference (Conventional)” on page 2-44.

- 8 Click **OK**. The Block Parameters dialog box closes.

- 9 Repeat the previous setup for the second RS-232 Setup block and the COM2 port. Use the same **Baudrate**, **Databits**, **Stopbits**, **Parity**, and **Protocol** that you entered in the first RS-232 Setup block.
- 10 Double-click the Send block. The Block Parameters dialog box opens.
- 11 From the **Port** list, select either COM1 or COM2. For this example, select COM1. In the **Message struct name** box, enter the name for the MATLAB structure this block uses to send messages to the COM1 port. In the box, enter the sample time or a multiple of the sample time you entered in the RS-232 Receive block.

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

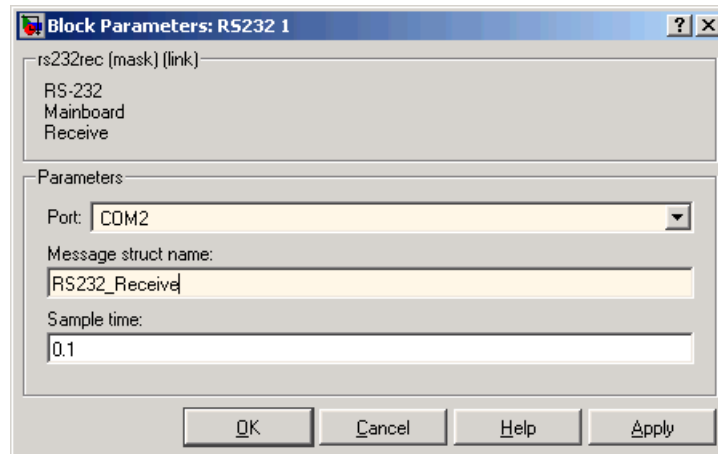


For information on entering the block parameters, see RS-232 Mainboard Send (Asynchronous) (Conventional). For the procedure to create the send structure, see “RS-232 MATLAB Structure Reference (Conventional)” on page 2-44.

- 12 Click **OK**. The Block Parameters dialog box closes.
- 13 Double-click the RS-232 Receive block.
- 14 The Block Parameters dialog box opens.

- 15 From the **Port** list, select either COM1 or COM2. For this example, select COM2. In the **Message Struct Name** box, enter the name for the MATLAB structure this block uses to receive messages from the COM2 port. In the **Sample Time** box, enter the sample time or a multiple of the sample time you entered in the RS-232 Send block.

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



For information on entering the block parameters, see RS-232 Mainboard Receive (Asynchronous) (Conventional). For the procedure to create the send structure, see “RS-232 MATLAB Structure Reference (Conventional)” on page 2-44.

- 16 Click **OK**. The Block Parameters dialog box closes.
- 17 Double-click the Signal Generator block and enter parameters. For example, from the **Wave Form** list, select sine. In the **Amplitude** and **Frequency** boxes, enter 1. From the **Units** list, select Hertz. Click **OK**.
- 18 Double-click the Gain block and enter parameters. For example, in the Gain box, enter -1. Click **OK**.

Your next task is to create the MATLAB message structures that the RS-232 driver blocks use to sequence commands to the RS-232 device. See “Creating RS-232 Message Structures (Synchronous)” on page 2-28.

### Creating RS-232 Message Structures (Asynchronous)

RS-232 drivers use MATLAB structures to send and receive messages and map the input and output ports on the RS-232 driver blocks to the data written and read from the RS-232 devices in synchronous mode.

After you add the RS-232 Setup, Asynchronous Send, and Asynchronous Receive blocks to your Simulink model, you can create the message structures to communicate with the RS-232 devices. You need to create and load these structures into the MATLAB workspace before you build your target application. The easiest way to create these structures is to use an M-file and load that M-file into the MATLAB workspace. See `xpcrs232v2.mdl` in the `xpcdemos` directory for an example model. That example sends and receives two floating-point numbers. In that example, both floating-point number fields for `SendData` are filled from `InputPorts 1` because only one input port is specified. In the case of `RecData`, the first floating-point number field is sent to `OutputPorts 1`, but the second floating-point number field is ignored because only one output port is specified.

The following procedure describes how to create an RS-232 message structure to send and receive one floating-point number:

- 1 In the MATLAB Command Window, and from the **File** menu, point to **New**, and then click **M-file**.

A MATLAB text editor window opens.

- 2 Enter the initialization, send, receive, and termination messages. Each message is an element in a MATLAB structure array with a series of fields. For information and examples of these fields, see “RS-232 MATLAB Structure Reference (Conventional)” on page 2-44.

For example, if you want to send and receive one floating-point number, type the following. In this example, the floating-point number field for `SendData` is filled from `InputPorts 1`. In the case of `RecData`, the floating-point number field is sent to `OutputPorts 1`.

---

**Note** Field names in the structures are case sensitive.

---

```
RS232_Send(1).SendData = 'start,%f,%f,stop;\r';
RS232_Send(1).InputPorts = [1];
RS232_Send(1).Timeout = 0.01;
RS232_Send(1).EOM = 1;
RS232_Receive(1).RecData = 'start,%f,%f,stop;\r';
RS232_Receive(1).OutputPorts = [1];
RS232_Receive(1).Timeout = 0.01;
RS232_Receive(1).EOM = 1;
```

---

**Note** If you do not manually load the message structures before opening your Simulink model, or have the message structures automatically loaded with the model, the port connections to the RS-232 blocks break.

---

If you want to send more than one variable, for example three, in a single frame, use the following `RS232_Receive(1).OutputPorts` line. This line sends the first %f data to output port 1, the second %f to output port 2, and the third %f to output port 3.

```
RS232_Receive(1).OutputPorts = [1 2 3];
```

**3** From the **File** menu, click **Save As**. In the Save As File dialog box, enter the name of the M-file. For example, enter

```
RS232Async_Messages.m
```

**4** Close the text editing window.

**5** In the MATLAB Command Window, type the name of the M-file you created with the RS-232 structures. For example, type

```
RS232Async_Messages
```

MATLAB loads and runs the M-file to create the message structures in the MATLAB workspace needed by the RS-232 driver blocks.

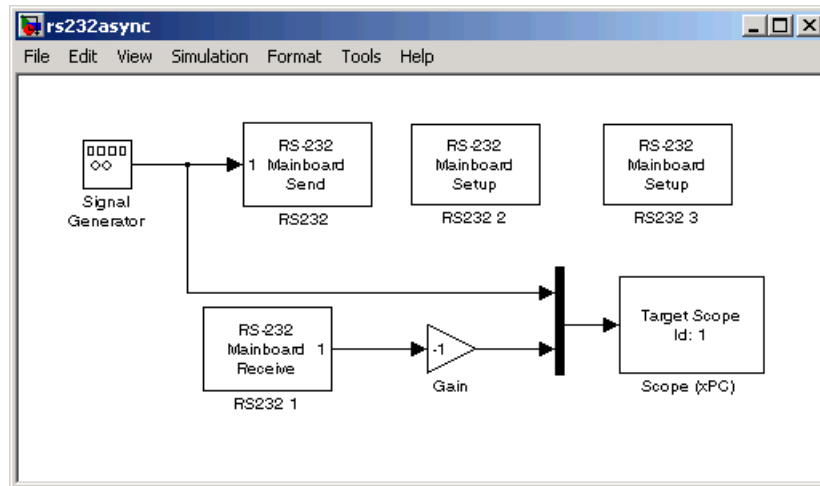
**6** Open your Simulink model, or press **Ctrl+D**.



Simulink updates the RS-232 driver blocks with the information from the structures. For example, Simulink adds the inputs and outputs defined in the structures to the blocks.

- 7 Connect the input and output ports on the RS-232 driver blocks to other blocks in your Simulink model.

Your model should look similar to the figure shown below.



- 8 Set the preload function for your Simulink model to load the message structures when you open the model. For example, if you saved the message structures in the M-file `RS232async_messages`, type

```
set_param(gcs, 'PreLoadFcn', 'RS232async_messages')
```

---

**Note** If you do not manually load the message structures before opening your Simulink model, or have the message structures automatically loaded with the model, the port connections to the RS-232 blocks break.

---

Your next task is to build and run the target application.

### **Building and Running the Target Application (Asynchronous)**

xPC Target and Real-Time Workshop create C code from your Simulink model. You can then use a C compiler to create executable code that runs on the target PC.

After you have added the RS-232 blocks for asynchronous mode to your Simulink model, and created and loaded the RS-232 structures into the MATLAB workspace, you can build your target application.

---

**Note** You cannot use a serial port to communicate between the host PC and target PC with this example. You can only use COM1 if it is not already in use for host-target communications.

---

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

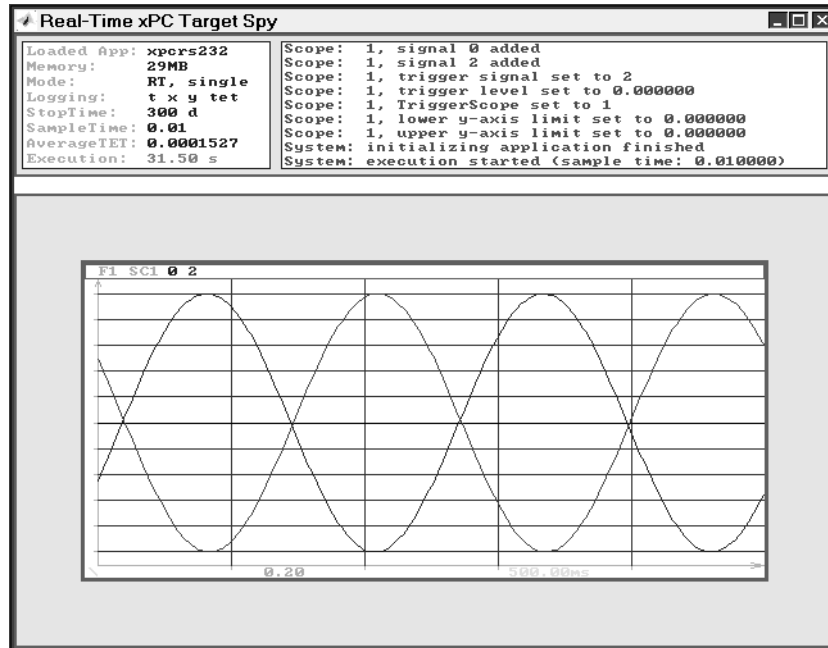
**2** In the MATLAB Command Window, type

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

The target application begins running in real time.

For each sample period, the RS-232 messages you entered in the RS-232 send and receive message structures are executed.

In this example, the target PC displays the inverted waveform. The RS-232 Send and RS-232 Receive blocks require a minimum delay or one sample to send the data and also receive it. When running at faster sample rates, several sample intervals might elapse while one set of data is transmitted, because RS-232 communication is not particularly fast. The sample delay just described is not visible in this example.



You can extend this example for multiple D/A channels by simply adding more input signals and modifying the format string to have additional '%f' format specifiers.

---

**Note** This example requires that you not use host PC to target PC communication using a serial port because that would block that COM port and the example would not operate.

---

## **RS-232 Simulink Block Reference (Conventional)**

xPC Target supports RS-232 communication with driver blocks in your Simulink model and message structures in the MATLAB workspace.

This section includes the following topics:

- **RS-232 Mainboard Setup (Conventional)** — Sends the initialize and termination messages. You need one Setup block for each RS-232 port you use in your model.
- **RS-232 Mainboard Send/Receive (Synchronous) (Conventional)** — Sequences the send and receive messages for synchronous serial communication.
- **RS-232 Mainboard Send (Asynchronous) (Conventional)** — Sequences the send messages.
- **RS-232 Mainboard Receive (Asynchronous) (Conventional)** — Sequences the receive messages.

## **RS-232 MATLAB Structure Reference (Conventional)**

You do not use all message fields in all messages. For example, a message to send data would not use the message field `.RecData`, but would use the field `.SendData`. However, knowing the possible message fields is helpful when you are creating any of the message structures. This section contains the following topics:

- **“RS-232 Send/Receive Message Structure (Synchronous)”** on page 2-45 — Description of the message fields for the send/receive structure associated with RS-232 asynchronous mode and the RS-232 Send/Receive block
- **“RS-232 Send Message Structure (Asynchronous)”** on page 2-46 — Description of the message fields for the send structure associated with RS-232 synchronous mode and the RS-232 Send block
- **“RS-232 Receive Message Structure (Asynchronous)”** on page 2-47 — Description of the message fields for the receive structure associated with RS-232 synchronous mode and the RS-232 Receive block
- **“Supported Data Types for Message Fields”** on page 2-47 — List of supported data types and the format you use to indicate those types in message fields

## RS-232 Send/Receive Message Structure (Synchronous)

Below are descriptions of the possible message fields for the send/receive structures with asynchronous mode. The order of the fields does not matter. However, the field names are case sensitive.

Message Field	Description
SendData	<p>Data and format sent to the RS-232 device. Default value = ' '.</p> <p>Note that the SendData syntax format is the same as the C <code>printf()</code> library function. It is also very similar to the MATLAB <code>fscanf</code> method, with the exception that SendData is not vectorized.</p>
InputPorts	<p>Number of input ports for the driver block. Data from the input ports is sent to the RS-232 device with the message field <code>.SendData</code>. Default value = []. The highest number you enter determines the number of input ports on the driver block</p> <p>For example, the following message creates two input ports on the driver block,</p> <pre>RS232_Send_Receive(1).InputPorts= [1 2];</pre>
RecData	<p>Data and format received from the RS-232 device. Default value = ' '. The format of this statement is very similar to a <code>scanf</code> statement. The read data is mapped to the output ports defined in the message field <code>.OutputPorts</code>. If a negative output port is given, the data is read in, but not sent to any output port.</p>
OutputPorts	<p>Number of output ports from the driver block. Data received from an RS-232 device is sent to the output ports with the message field <code>.RecData</code>. Default value = []. The highest number you enter determines the number of output ports on the driver block.</p> <p>For example, to use output ports 1 and 2 on the driver block, type</p> <pre>RS232_Send_Receive.OutputPorts = [1 2];</pre>

Message Field	Description
Timeout	Time, in seconds, the driver block waits for data to be returned. Default value = 0.049.
EOM	Number of characters you use to indicate the end of a message.

**RS-232 Send Message Structure (Asynchronous)**

Below is a description of the possible message fields for the send structure with synchronous mode. The order of the message fields does not matter. However, the field names are case sensitive.

Message Field	Description
SendData	Data and format sent to the RS-232 device. Default value = ' '.  Note that the SendData syntax format is the same as the C printf() library function. It is also very similar to the MATLAB fscanf method, with the exception that SendData is not vectorized.
InputPorts	Number of input ports for the driver block. Data from the input ports is sent to the RS-232 device with the message field .SendData. Default value = []. The highest number you enter determines the number of input ports on the driver block.  For example, the following message creates two input ports on the driver block.  <code>RS232_Send_Receive(1).InputPorts= [1 2];</code>
Timeout	Time, in seconds, the driver block waits for data to be returned. Default value = 0.049.
EOM	Number of characters you use to indicate the end of a message.

## RS-232 Receive Message Structure (Asynchronous)

Below are descriptions of the possible message fields for the receive message structure with synchronous mode.

Message Fields	Description
RecData	Data and format received from the RS-232 device. Default value = ''. The format of this statement is very similar to a scanf statement. The read data is mapped to the output ports defined in the message field .OutputPorts. If a negative output port is given, the data is read in but not sent to any output port.
OutputPorts	Number of output ports from the driver block. Data received from an RS-232 device is sent to the output ports with the message field .RecData. Default value = []. The highest number you enter determines the number of output ports on the driver block.  For example, to use output ports 1 and 2 on the driver block,  <pre>RS232_Send_Receive.OutputPorts = [1 2];</pre>
Timeout	Time, in seconds, the driver block waits for data to be returned. Default value = 0.049.
EOM	Number of characters you use to indicate the end of a message.

## Supported Data Types for Message Fields

The following table lists the supported data types for the RS-232 message fields.

Format	Description
%c and %C	Single character
%d or %i	Signed decimal integer
%u	Unsigned decimal integer

<b>Format</b>	<b>Description</b>
%o	Unsigned octal integer
%x or %X	Unsigned hexadecimal integer using 'abcdef' or 'ABCDEF' for the hexadecimal digits
%e or %E	Exponential format using e or E
%f	Floating point
%g	Signed value printed in f or e format depending on which is smaller
%G	Signed value printed in f or E format depending on which is smaller

## **RS-232 Binary Mode (Conventional)**

Use RS232 Binary mode when you want to transfer raw data. The format of this data is either a custom format or is an image of the bytes as they are stored in memory. This section includes the following topics:

- “RS-232 Binary Mode I/O” on page 2-48 — When to use RS-232 binary mode
- “RS-232 Binary Mode I/O” on page 2-48 — How to select drivers from the xPC Target block library
- RS-232 Binary Receive (Conventional) — Explanation of block parameters, inputs, and outputs
- RS-232 Binary Send (Conventional) — Explanation of block parameters and input
- “Example Using RS-232 Binary Mode I/O” on page 2-51 — Simulink model using xPC Target driver blocks

## **RS-232 Binary Mode I/O**

The binary mode drivers operate in asynchronous mode. In other words, they do not wait until an entire packet of data is received, but receive as many bytes as available and then go on to the next data block. When an entire packet has been received, the block outputs the new data. Sent data is also handled similarly. The Send block instructs the RS-232 hardware to send a certain number of bytes, but does not wait for these bytes to actually be sent.



The RS-232 binary mode infrastructure also includes blocks to pack and unpack any data received. This translates the raw bytes into signals that Simulink can understand.

The functioning of these blocks is identical to the corresponding blocks in the UDP section of the xPC Target block library. The RS232 Binary Pack and Unpack blocks are actually references to these blocks. For information about UDP and the functionality of these blocks, see Chapter 6, “UDP I/O Support”

### Using RS-232 Binary Mode

To use the RS-232 binary mode blocks, you must first insert exactly one RS232 Setup block for each COM port into your model. The setup for this block is exactly the same as it is for text-based I/O, except that initialization or termination structures are ignored. In the dialog box, set both these fields to the empty matrix ( []).

The RS-232 binary mode blocks can be found in the RS-232 section of the xPC Target Block Library. Use the following procedure to access these blocks:

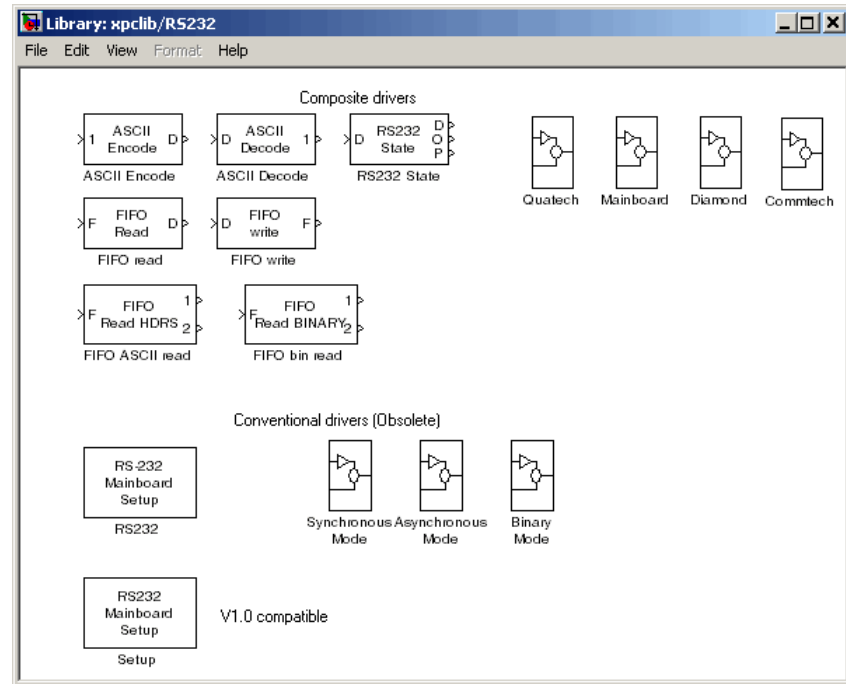
- 1 In the MATLAB Command Window, type

```
xpclib
```

The xPC Target Block Library opens.

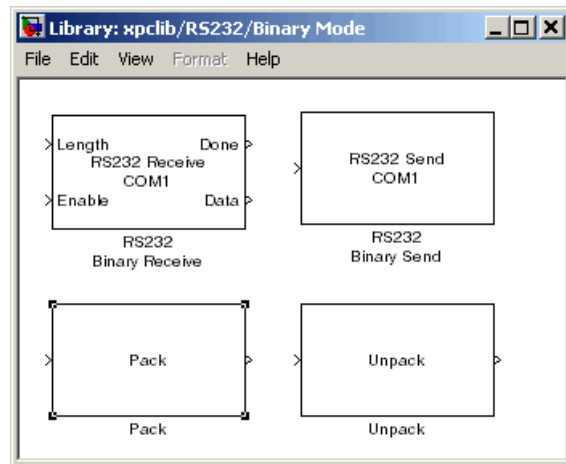
- 2 Double-click the group block **RS232**.

The **Library: xplib/RS232** library opens.



3 Double-click the group block **Binary Mode**.

The **Library: xplib/RS232/Binary Mode** library opens.



**4** Drag and drop any of these blocks into your Simulink model.

### Example Using RS-232 Binary Mode I/O

To show the flexibility provided by the RS232 Binary Receive block, the following illustrates how this block can be used. The model that implements this setup is provided with xPC Target. To access this model, type

```
xpcrs232bindemo
```

at the command prompt. This opens the model, which is essentially self-documenting. Open each subsystem in the model to see what that part is supposed to accomplish.

Here is an example of a messaging protocol that the model has to conform to.

The protocol consists of a one-byte header, followed by a variable-length body. The header can have only two legal values, 12 and 17. If the header is 12, the body is 6 bytes long, and consists of a uint16 followed by an int32 (in terms of MATLAB data types). If the header is 17, the body is 4 bytes long, and consists of a uint16 followed by an int16.

The model receives one header byte at a time, rejecting any invalid ones. As soon as a valid header byte is received, the execution switches to the body block, where the proper number of bytes is received. The data is then appropriately decoded and displayed on an xPC Target Scope of type target. The model should serve as an example of how this is done.

The basic algorithm is to receive a header byte and then compare it to the list of known headers (12 and 17). The body length is set appropriately depending on the header, and the “Done” function-call output of the header block is used to trigger functioning of the body block (via the “distributor“ function call subsystem).

## **Boards and Blocks – Alphabetical List**

# ASCII Encode/Decode

---

<b>Purpose</b>	ASCII Encode/Decode
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	<p>The ASCII Encode block generates a UINT8 output vector that contains a NULL-terminated string based on a printf like format string and data on the input ports.</p> <p>The ASCII Decode block parses an input vector according to a format specifier similar to scanf, and makes converted values available to a Simulink program. The input vector to the ASCII Decode block can be either 8-bit or 16-bit and signed or unsigned. If the data format is 16-bit, the ASCII Decode block ignores the upper eight bits of each entry.</p>
<b>Encode Block Parameters</b>	<p><b>Format string</b> Enter a printf like format string. Each format specifier such as %d is replaced by the converted value that is present on the corresponding input variable. Acceptable format specifiers are %c, %d, %i, %o, %u, %x, %e, %f, and %g. These follow the normal description for printf.</p> <p><b>Number of variables</b> Enter the number of input ports to this block. The value on each port is inserted into the output string with the format specified in <b>Format string</b>.</p> <p><b>Max output string length</b> Enter the maximum allowed length of the converted string, in bytes. The block allocates enough memory to support this length for the output port. When selecting this length, take into account the NULL termination on the string.</p> <p>If the converted string exceeds this length, the block returns an error and does not write that string to the output port.</p> <p><b>Variable types</b> Enter one of the following: {'double'}, {'int8'}, {'uint8'}, {'int16'}, {'uint16'}, {'int32'}, and {'uint32'}. The default</p>

is `{'double'}`. This parameter specifies the Simulink data types allowed for the input ports. A cell vector with the same number of elements as specified in **Number of variables** can specify a different data type for each input port. A single element is replicated. For example,

```
nvars=3
```

`{ }` — All three inputs are doubles.

`{'uint8'}` — All three inputs are uint8.

`{'uint16', 'double', 'uint8'}` — There are three inputs: the first is a uint16, the second is a double, and the third is a uint8.

## Decode Block Parameters

### Format string

Enter a scanf like format string. Each format specifier such as `%d` needs to match a corresponding part of the input vector. Literal strings in the format need to match the first character plus the number of characters. Acceptable format specifiers are `%c`, `%d`, `%i`, `%o`, `%u`, `%x`, `%e`, `%f`, and `%g`. These follow the normal description for `scanf`.

### Number of variables

Enter the number of output ports for this block. For example,

If **Format string** has the value of `%xmore text%x` and the input vector for the block has `cdmabcde fgh90`, you must specify the value of the **Number of variables** parameter as 2.

The first variable is assigned the value `0xcd`. Next, the string `mabcde fgh` is considered a match to `more text` because

- The first character for both strings is `m`.
- Both strings have the same number of characters.

The second variable is then assigned the value `0x90`. Note that the string `mabcde fgh` does not have to match exactly the value of

**Format string.** This behavior is different from that for `scanf`, which requires an exact match.

## Variable types

Enter one of the following: `{'double'}`, `{'int8'}`, `{'uint8'}`, `{'int16'}`, `{'uint16'}`, `{'int32'}`, and `{'uint32'}`. The default is `{'double'}`. This parameter specifies the Simulink data types allowed for the output ports. A cell vector with the same number of elements as specified in **Number of variables** can specify a different data type for each output port. A single element is replicated. For example,

```
nvars=3
```

`{ }` — All three outputs are doubles.

`{'uint8'}` — All three outputs are `uint8`.

`{'uint16', 'double', 'uint8'}` — There are three outputs: the first is a `uint16`, the second is a `double`, and the third is a `uint8`.



**Purpose** FIFO Read/Write

**Library** xPC Target Library for RS232

**Description** This section describes the FIFO Read and Write blocks. Of particular note are the FIFO Read blocks, of which there are three. Use the following guidelines when using these blocks:

- Simple data streams — Use the FIFO Read block to read simple data streams. An example of a simple data stream is one that has numbers, separated by spaces, and that is terminated by a newline. The FIFO Read block is a simple block that can easily extract these numbers.
- More complicated data streams — Use the FIFO Read HDRS and FIFO Binary blocks for more complicated data streams. A more complicated data stream can be one that contains headers, messages of varying lengths, or messages with no specific terminators. A message header consists of one or more character identifiers at the beginning of a message that specify what data follows. ASCII messages normally have a variable length and a terminator. Typically, all messages of a particular device use the same predefined terminator. Binary messages are normally of fixed length with no specific terminator.

The FIFO Read HDRS or FIFO Binary blocks are also useful to work with devices that can send different messages at different times.

All three FIFO read block types need their input to be of type `serialfifoPtr`, which is output from F type Send Receive subsystems.

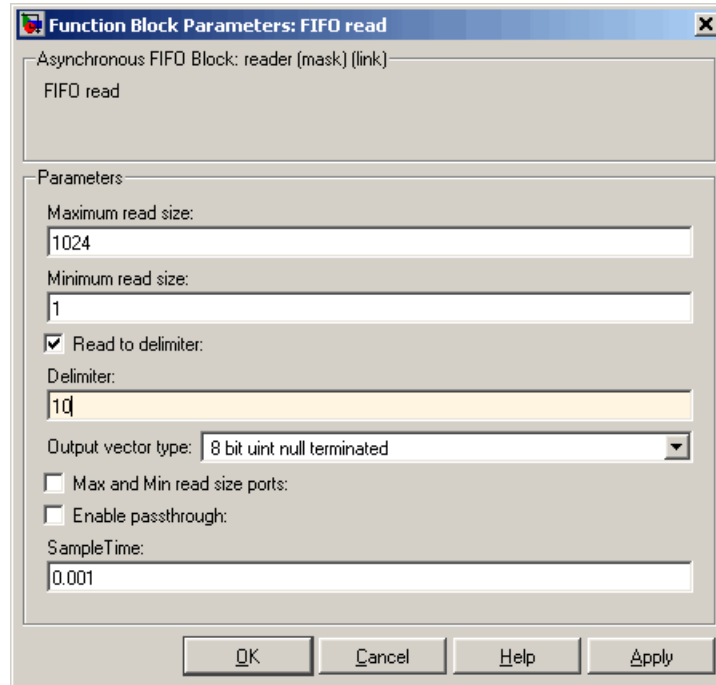
The following are examples of when you can use the FIFO read block.

- For an instrument that sends a string like the form

```
<number> <number> ... <CR><LF>
```

# FIFO Read/Write

use the simple FIFO Read block to read the message. Configure the FIFO Read block **Delimiter** parameter for a line feed (value of 10).

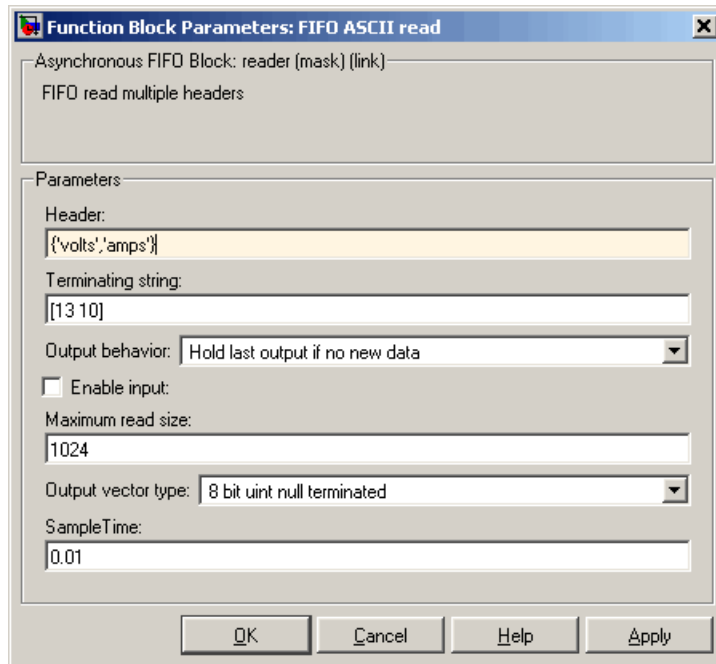


You should then connect the output to an ASCII Decode block with a format that separates the numbers and feeds them to the output ports.

- For an instrument that can send one of a number of different messages, and each message begins with a different fixed string, use the FIFO Read HDRS block. For example, a digital multimeter connected through an RS-232 port, might send a voltage reading and an amp reading with messages of the following format:

```
volts <number> <CR><LF>  
amps <number> <CR><LF>
```

Configure the FIFO Read HDRs block **Header** parameter for the volts and amps headers, in a cell array. Also configure the **Terminating string** parameter for carriage return (value of 13) and line feed (value of 10).



You should then connect the output to multiple ASCII Decode blocks, one for each header and message. See the `xpcserialasciitest` and `xpcserialasciisplit` models in `xpcdemos` for examples of how to use this block in a model.

- For an instrument that sends a binary message, you typically know the length of each full message, including the header. Configure the FIFO Read Binary block **Header** parameter for the headers of the message, in a cell array, and the **Message Lengths** parameter for the message lengths. See the `xpcserialbinarytest` and

# FIFO Read/Write

---

xpcserialbinarysplit models in xpcdemos for further examples of how to use this block in a model.

The FIFO Read block is the read side of a FIFO read/write pair. There are two modes for this block:

- If **Read to delimiter** is checked, this block only reads elements if the chosen delimiter is found in the FIFO. If the delimiter has not yet been written to the write side of this FIFO, the block returns a zero length vector, as determined by the data type. If the delimiter is found, the block returns elements up to and including the delimiter in the output vector. Selecting this check box cause the block to perform ASCII reads.
- If **Read to delimiter** is not checked, this block returns a number of elements between **Minimum read size** and the smaller of the number of elements currently in the FIFO and **Maximum read size**. Selecting this check box cause the block to perform binary reads.

The FIFO Write block is the write side of a FIFO read/write pair.

## FIFO Read Block Parameters

### Maximum read size

Specify the maximum number of characters that you will ever expect to be returned by this block. The resulting vector size will be one more than this maximum number of characters. This block needs the extra element to contain the number of characters that are being returned. The block contains the characters as either

- A null terminator for the 8-bit data types
- The character count for the 16- and 32-bit data types

Be sure to enter a large enough number. If this number is too small, the block might not be able to return anything. If this number is larger than necessary, there is no effect. For example, if you enter a value 10, but on execution there are 11 characters up to the terminator in the FIFO, the block will not return any characters.

If you select the parameter **Max and Min read size ports**, the block interprets the value input on this port to be the maximum number of characters to return. The actual maximum number of characters to return is the smaller of the value on the max input port or the maximum read size in the block parameters. This is mainly useful in binary mode when the **Read to delimiter** check box is not selected.

### **Minimum read size**

Enter the smallest desired read size in bytes. The FIFO must contain at least this number of elements before any elements will be returned. If you select the **Max and Min read size ports** check box, this value is superseded by the external signals.

### **Read to delimiter**

Select this check box to enable the return of element sets that terminate with the **Delimiter** value. Use this parameter when working with character-based elements.

### **Delimiter**

Enter the decimal value for an 8-bit input terminator. This parameter specifies the value on which a FIFO read operation should terminate. It works with the **Read to delimiter** parameter. By default, this block looks for a carriage return. It only returns characters when one is found. For reference, the decimal value of a carriage return is 13, a line feed is 10.

### **Output vector type**

From the list, select count+32 bit int, count+32 bit uint, count+16 bit int, count+16 bit uint, 8 bit int null terminated, or 8 bit uint null terminated. This parameter specifies the output vector type. The 8-bit data types produce a null terminated string in the output vector. For 16- and 32-bit data types, the first element contains the number of elements to expect in the rest of the output vector.

### **Max and Min read size ports**

Select this check box to enable the maximum and minimum input ports. When this check box is selected,

# FIFO Read/Write

---

The value from the maximum input port is the maximum number of characters to be removed from the FIFO. Note that if this number exceeds the value of **Maximum read size**, the block disregards the value from the maximum input port and takes the value of **Maximum read size** as the maximum number of characters to be removed from the FIFO.

The value from the minimum input is the minimum number of characters the FIFO must contain before any elements can be returned. This value supersedes the value set with the **Minimum read size** parameter.

## **Enable passthrough**

Select this check box to pass the maximum read input through to the passthrough output.

## **SampleTime**

Base sample time or a multiple of the base sample time.

## **FIFO Write Block Parameters**

### **Size**

Enter the number of elements that can be held in the FIFO at any one time. If a write operation to the FIFO causes the number of elements to exceed **Size**, an error occurs.

### **Input vector type**

From the list, select count+32 bit int, count+32 bit uint, count+16 bit int, count+16 bit uint, 8 bit int null terminated, or 8 bit uint null terminated. This parameter specifies the input vector type. The 8-bit data types need a null terminated string in the output vector. For 16- and 32-bit data types, the first element contains the number of elements to expect in the rest of the input vector.

### **Data present output**

Select this check box to create a Boolean output that is true if data is present in the FIFO. The transmit side of the send/receive subsystem uses this output. This output is given to the Enable TX block, which enables the transmitter buffer empty interrupt.

**SampleTime**

Base sample time or a multiple of the base sample time.

**ID**

Enter a user-defined identifier for overflow messages.

**Example**

The following are some examples of how you can set up the FIFO Read block:

- In the transmit side of the interrupt service routine, the maximum input port receives a value of 0 if the interrupt reason is not an empty hardware FIFO, and the hardware FIFO size if the hardware FIFO is empty. The minimum input port receives the constant value of 1.

On the receive side, the typical case with ASCII data has the minimum and maximum input ports disabled. The **Read to delimiter parameter** check box is selected and the **Delimiter** parameter has the value of carriage return or line feed. The value of the **Maximum read size** parameter is large (along the order of the FIFO size) and the value of **Minimum read size** parameter is 1. In this form, the driver acts like a nonblocking read line.

- An alternate receive-side configuration for fixed-length binary blocks of data has the value of the **Maximum read size** and **Minimum read size** parameters set to the fixed length of the block. The **Read to delimiter** parameter is not selected.

# FIFO Read HDRS

---

**Purpose** FIFO Read HDRS block

**Library** xPC Target Library for RS232

**Description** The FIFO Read HDRS block identifies and separates ASCII data streams that have embedded identifiers. The data following a particular header might have varying lengths, but all have a common termination marker such as <CR><LF>. While you can attain this same functionality with the sample FIFO Read block, doing so requires a complicated state machine.

The xpcdemos directory contains the following demos that illustrate how to use the FIFO Read HDRS block: xpcserialasciitest and xpcserialasciisplit.

## Block Parameters

### Header

Enter the headers that you want the block to look for in a block of data from the FIFO. Enter each header as an element in a cell array.

### Terminating string

Enter the terminating string for the data. Enter as many characters as needed to define the end of string. This is typically one or two characters.

### Output behavior

From the list, select the behavior of the block if the FIFO has no new data. Select **Zero output if no new data** if you want the block to have no output if there is no new data. Select **Hold last output if no new data** if you want the block to keep the output from the last FIFO message.

### Enable input

Select this check box to turn on a new input that takes Boolean signals that enable or disable the read.

### Maximum read size

Enter the largest desired read size in bytes. This parameter specifies the width of the output vector and the maximum



number of elements to return. See **Output vector type** for more information about data formats.

### **Output vector type**

From the list, select count+32 bit int, count+32 bit uint, count+16 bit int, count+16 bit uint, 8 bit int null terminated, or 8 bit uint null terminated. This parameter specifies the output vector type. The 8-bit data types produce a null terminated string in the output vector. For 16- and 32-bit data types, the first element contains the number of elements to expect in the rest of the output vector.

### **SampleTime**

Base sample time or a multiple of the base sample time.

# FIFO Read Binary

---

**Purpose** FIFO Read Binary

**Library** xPC Target Library for RS232

**Description** The FIFO Read Binary block reads multiple binary headers from a FIFO. This block identifies and separates data by finding unique byte sequences (headers) that mark the data. Each header indicates the start of a fixed length binary message.

The xpcdemos directory contains the following demos that illustrate how to use the FIFO Read HDRS block: xpcserialbinarytest and xpcserialbinarysplit.

## Block Parameters

### Header

Enter the headers that you want the block to look for in a block of data from the FIFO. Enter each header as an element in a cell array either as a quoted string or a concatenation with `char(val)` for non-printable byte patterns.

### Message Lengths

Enter the message length of each byte of data as bytes of data. Include the header in the length.

### Output behavior

From the list, select the behavior of the block if the FIFO has no new data. Select **Zero output if no new data** if you want the block to have no output if there is no new data. Select **Hold last output if no new data** if you want the block to keep the output from the last FIFO message.

### Enable input

This check box enables or disables a FIFO read. Select this check box to turn on a new input that takes Boolean signals that enable or disable the read.

### Maximum read size

Enter the largest desired read size in bytes. This parameter specifies the width of the output vector and the maximum

number of elements to return. See **Output vector type** for more information about data formats.

## **Output vector type**

From the list, select count+32 bit int, count+32 bit uint, count+16 bit int, count+16 bit uint, 8 bit int null terminated, or 8 bit uint null terminated. This parameter specifies the output vector type. The 8-bit data types produce a null terminated string in the output vector. For 16- and 32-bit data types, the first element contains the number of elements to expect in the rest of the output vector.

## **SampleTime**

Base sample time or a multiple of the base sample time.

# Modem Control

---

**Purpose** Modem Control block

**Library** xPC Target Library for RS232

**Description** The Modem Control block controls the state of either or both of the RTS and DTR output lines on the specified port.

This block requires an input of type `double`. If the input value is greater than 0.5, the block asserts the RTS or DTR control bit to true and the output goes to a positive voltage. If the value is less than or equal to 0.5, the block asserts the RTS or DTR control bit to false and the output goes to a negative voltage. If RTS or DTR is not selected, the corresponding output is not changed.

**Port** (Quatech, Commtech)

From the list, choose a port. The **Port** parameter defines the port to configure for this driver block.

**RTS**

Select this check box to control the RTS line for this board.

**DTS**

Select this check box to control the DTR line for this port.

**Port** (Diamond)

From the list, choose a port. The **Port** parameter defines the port to configure for this driver block.

**First port address** (Diamond)

For Emerald-MM, this value should be the same as the **First port address** parameter value you select in the Parameter Group: Board Setup dialog of the Send/Receive block. See the Diamond user's manual documentation for the appropriate jumper settings.

For Emerald-MM-8, this parameter contains a value based on the **Base address** value of the configuration register in the Parameter Group: Board Setup dialog of the Send/Receive block. See the Diamond user's manual documentation.

## **Configuration** (Mainboard)

From the list, choose a port. This parameter specifies the port whose input modem control line states you want to read.

Normally, the ports are set to the following:

COM1 — 0x3F8

COM2 — 0x2F8

COM3 — 0x3E8

COM4 — 0x2E8

A Custom port is one that is set to an address other than these.

## **Slot (PCI boards)**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Modem Status

---

**Purpose** Modem Status block

**Library** xPC Target Library for RS232

**Description** The Modem Status block reads the state of the four input modem control lines.

This block has an output of type `Boolean`. If the input voltage is positive, the output is true. If the input voltage is negative, the output is false.

**Port** (Quatech, Commtech)

From the list, choose a port. The **Port** parameter defines the port to configure for this driver block.

**CTS**

Select this check box to monitor the CTS line.

**DSR**

Select this check box to monitor the DSR line.

**RI**

Select this check box to monitor the RI line.

**DCD**

Select this check box to monitor the DCD line.

**Sample Time**

Base sample time or a multiple of the base sample time.

**Port** (Diamond)

From the list, choose a port. The **Port** parameter defines the port to configure for this driver block.

**First port address** (Diamond)

This parameter specifies the first port whose input modem control line states you want to read. Do not change this value.

For Emerald-MM, this value should be the same as the **First port address** parameter value you select in the Parameter Group: Board Setup dialog.

For Emerald-MM-8, this parameter contains a value based on the **Base address** value in the Parameter Group: Board Setup dialog.

## **Configuration (Mainboard)**

From the list, choose a port. This parameter specifies the port whose input modem control line states you want to read.

Normally, the ports are set to the following:

COM1 — 0x3F8

COM2 — 0x2F8

COM3 — 0x3E8

COM4 — 0x2E8

A Custom port is one that is set to an address other than these.

## **Slot (PCI boards)**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# RS-232/422/485 Board Setup (Commtech) and Interrupt Check (Quatech) (Composite)

---

**Purpose** RS-232/422/485 Board Setup (Commtech) and Interrupt Check (Quatech) block

**Library** xPC Target Library for RS232

**Description** (Quatech and Commtech only) The Board Setup and Interrupt Check block checks for instances where the hardware IRQ differs from the software for which it is listening. This block compares the software-selected interrupt against the value for which the board (PCI only) is configured. This check prevents IRQ mismatches. For the Commtech Fastcom board, the corresponding block also sets up the board clock.

## **Block Parameters**

### **PCI Slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

### **Clock Bits (Commtech)**

Enter the number of clock bits to control a clock generator common to both channels. This parameter adjusts the master clock of both channels.

### **IRQ Line Number**

From the list, select an IRQ line number from 5 to 15, inclusive.



# RS-232/422/485 Enable TX Interrupt (Composite)

---

<b>Purpose</b>	RS-232/422/485 Enable TX Interrupt block
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	<p>The Enable TX Interrupt block enables the transmitter buffer empty interrupt when data is present in the software FIFO.</p> <p>The input port for controlling this is a Boolean value. If the input port value is true, the Enable Transmit Interrupt block enables the transmitter buffer empty interrupt in the UART. After the interrupt service routine empties the software FIFO, the interrupt is disabled.</p>
<b>Block Parameters</b>	<p><b>Base address</b> (Mainboard, ISA boards) Enter the base address of the UART for which you want to enable the transmitter buffer empty interrupt.</p> <p><b>Port</b> (Quatech, Commtech) From the list, choose a port. This parameter specifies the input port for which this block enables the interrupt.</p> <p><b>PCI Slot (PCI boards)</b> If only one board of this type is in the target PC, enter</p> <p style="padding-left: 40px;">-1</p> <p>to automatically locate the board.</p> <p>If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type</p> <p style="padding-left: 40px;">getxpcpci</p>

# RS-232/422/485 Filter Interrupt Reason (Composite)

---

**Purpose** RS-232/422/485 Filter Interrupt Reason block

**Library** xPC Target Library for RS232

**Description** The Filter Interrupt Reason block filters the output of the Read Int(errupt) Status block. If the condition that the interrupt query block reads from the IIR register matches the one chosen here, the output is true.

This block is used exclusively inside the interrupt service subsystem for this board.

**Block Parameters**

**Port**  
From the list, choose a port. This parameter specifies the port from which this block gets control data.

**Filter value**  
From the list, choose Receive data, Transmitter empty, or Modem status change. This parameter specifies the interrupt reason that this filter block is looking for.

Note that Modem status change currently has no effect because the interrupt is never enabled.

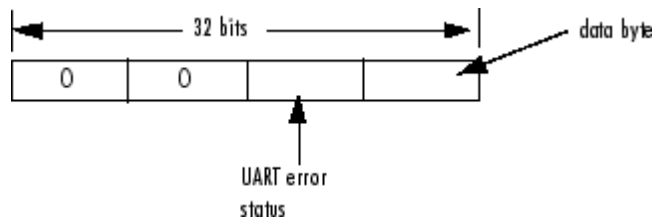
# RS-232/422/485 Read Hardware FIFO (Composite)

**Purpose** RS-232/422/485 Read Hardware FIFO block

**Library** xPC Target Library for RS232

**Description** The Read Hardware FIFO block reads characters from the hardware FIFO in the UART. It then outputs those characters as the low-order byte of an unsigned 32-bit integer vector with a width of 65. This output vector is large enough to hold the maximum number of characters that the hardware FIFO can hold. The first element of the vector specifies the number of data elements in the remainder of the vector.

If the input to the enable port (input port, labeled E) is not true, this block outputs a 0 length vector. The following illustrates the vector.



The UART error status can contain one of the following error values:

- 0x02 — Overrun error
- 0x04 — Parity error
- 0x08 — Framing error
- 0x01 — Break interrupt

The data byte ranges from 0 to 255.

The dialog box for the RS-232 FIFO Read block contains the following fields:

**Block Parameters**

**Port** (Quatech, Commtech)

From the list, choose a port. This block reads the hardware FIFO from this port.

# RS-232/422/485 Read Hardware FIFO (Composite)

---

## **Flush HW FIFO on startup**

Select this check box to flush the hardware FIFO when the device starts up.

## **Base address (Mainboard, ISA boards)**

Enter the base address of the UART for which you want to read the hardware FIFO.

## **PCI Slot (PCI boards)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# RS-232/422/485 Read Int(errupt) Status (Composite)

**Purpose** RS-232/422/485 Read Int(errupt) Status block

**Library** xPC Target Library for RS232

**Description** The Read Interrupt Status block reads the interrupt status for the boards in the system. The output for this block is a vector with one 32-bit element for each port. Each element contains two pieces of information for that port, where the four bytes are

[0, 0, IIR, Reason]

The Read Interrupt Status block has signal output with the following format:

This output is a vector of integers. The values in the reason byte and their definitions are

0 — This UART did not cause this interrupt.

1 — Receive characters are available.

2 — Transmit holding register is empty.

3 — Modem status has changed (ignored).

This second byte is the value read from the Interrupt Reason Register (IIR). This register is specific to the 16450, 16550, and 16750 types of UARTs. Several bites in this register indicate the active hardware FIFO depth and the maximum number of characters that can be written in the transmitter empty interrupt handlers to the transmit hardware FIFO.

## **Block Parameters**

**Base address 1** (Mainboard, ISA boards)

Enter the base address of the first UART for which you want to read the interrupt status.

**Base address 2** (Mainboard, ISA boards)

Enter the base address of the second UART for which you want to read the interrupt status.

# RS-232/422/485 Read Int(errupt) Status (Composite)

---

## PCI Slot (PCI boards)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## First to Fourth port address (Diamond-MM)

These parameters contain the address of the port for which you want to read the interrupt status.

Starting from the **First port address** parameter value you select in the Parameter Group: Board Setup dialog of the Send/Receive block, the subsequent parameters contain the incremental port addresses.

## First to Eighth port address (Diamond-MM-8)

These parameters contain the addresses of the ports for which you want to read the interrupt status.

Starting from the **Base address** parameter value in the Parameter Group: Board Setup dialog of the Send/Receive block, the subsequent parameters contain the incremental port addresses.

## Interrupt status address (Diamond)

This parameter contains the address for the interrupt status.

This parameter derives from the configurations you define in the Parameter Group dialog of the Send/Receive block.

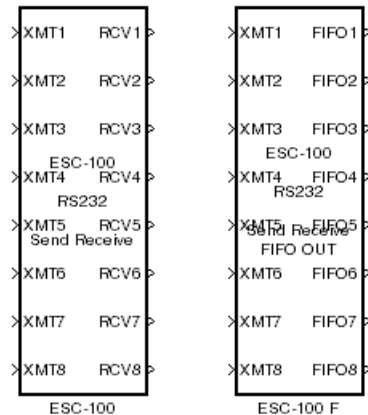
# RS-232/RS-422/RS-485 Send/Receive

**Purpose** RS-232/RS-422/RS-485 Send/Receive block

**Library** xPC Target Library for RS232

**Description** The dynamic dialog for these subsystem blocks allows you to perform basic board setup and setup of send/receive data. You control parameter visibility with the **Parameter Group** parameter and the port number.

There are two versions of this block, non-F and F (FIFO). All serial boards plus the main board have these two versions. The primary difference is that the F blocks bring the FIFO signal out of the subsystem. For example



- The non-F blocks have RCV outputs. These blocks have basic FIFO read blocks inside the subsystem. These blocks are most useful for simple character streams. These subsystem blocks generate output as an array of packed integers (settable at 8, 16, or 32 bits) with characters in the lower byte and received status information in the upper byte.
- The F blocks have FIFO outputs. These blocks give you greater flexibility and allow you to use any one of the FIFO read blocks.

# RS-232/RS-422/RS-485 Send/Receive

---

- FIFO Read block — A model that contains an F block in combination with the FIFO Read block provides the same capability as the non-F block.
- FIFO Read HDRS and FIFO Read Binary — A model that contains an F block in combination with a FIFO Read HDRS or FIFO Read Binary block provides greater capability than the FIFO Read block. (See FIFO Read HDRS and FIFO Read Binary for details.)

Only one Send/Receive block can exist for each interrupt. All ports that use that interrupt must be associated with that block.

For example, if you have four ports configured on the mainboard, COM1 and COM3 typically share an interrupt. In this case, COM1 and COM3 must then share the same Send/Receive block. COM1 is also of note because you can use it for host PC/target PC communication. If COM1 is the host PC/target PC link, neither COM1 nor COM3 can be used with this block as long as they share an interrupt. The same is true for COM2 and COM4.

Each F Send Receive has Board Setup and Basic Setup configuration parameter options that are the same as their non-F Send Receive block counterparts. To provide direct access to the board, these blocks also have a FIFO Setup parameter option. The following descriptions do not distinguish between the F and non-F blocks unless necessary.

## Block Parameters

The **Parameter Group** parameter allows you to choose which subset of configuration parameters you want to modify.

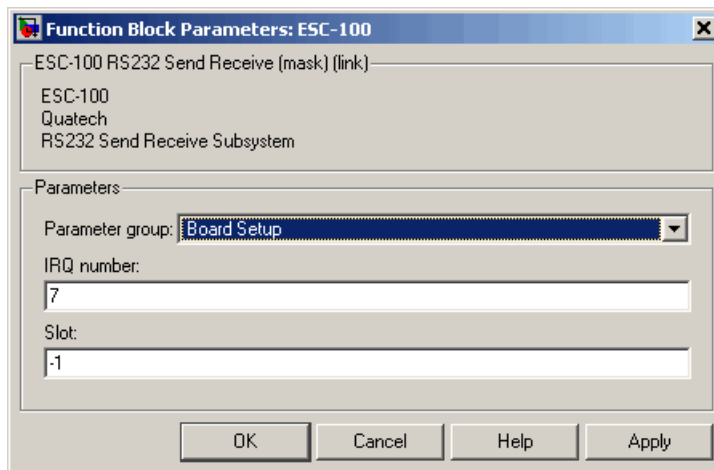
### Parameter Group

From the list, choose Board Setup, Basic Setup, Transmit Setup, or Receive Setup. The visible set of parameters changes according to your selection. The following screenshots reflect the Send Receive block for the Qatech ESC-100.



# RS-232/RS-422/RS-485 Send/Receive

Parameter Group: Board Setup



## Configuration (Mainboard)

From the list, choose combinations of port pairs (Com1 / none, Com2 / none, Com1 / Com3, Com2 / Com4, none / Com3, none / Com4, or Custom). This parameter specifies the ports for which you are defining transmit and receive. A Custom port is one that does not match the existing combinations of port pairs. For example, you can set the IRQ and two addresses or, if one of the ports is not used, set that to 0.

## IRQ number (Quatech, Commtech)

Enter the number of the interrupt request line for this board. If you do not know the interrupt request line number for this board, at the MATLAB Command Window, enter

```
getxpcpci
```

This command displays all the PCI interfaces currently attached to the target PC. From that display, find the instance of the board controlled by this block. Each board uses a unique interrupt request line number.

# RS-232/RS-422/RS-485 Send/Receive

---

For Emerald-MM, set this to the IRQ chosen by jumpers on the board. Set all four ports to the same IRQ.

For Emerald-MM-8, this block setting programs the board IRQ.

(Diamond) Not all IRQ values work in all target PC machines. You should experiment to find a working combination.

## **Clock Bits** (Commtech)

Enter the number of click bits to control a clock generator common to both channels. This parameter adjusts the master clock of both channels.

## **Slot (PCI boards)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## **Base address** (Diamond Emerald-MM-8)

Enter the base address of the board that you are setting up. This is the address of the configuration register on the board. The first port address referenced in Modem Control and Modem Status blocks is offset 0x10 from the configuration register address, and each subsequent port address is offset 0x8 from that.

You must set the configuration register address on the board with a jumper. You can set the configuration register address to one of the following addresses: 0x100, 0x140, 0x180, 0x1c0, 0x200, 0x240, 0x280, 0x2c0, 0x300, 0x340, 0x380, or 0x3c0. Note

that xPC Target assigns successive 8 byte addresses to the eight UARTs even though the hardware allows random placement.

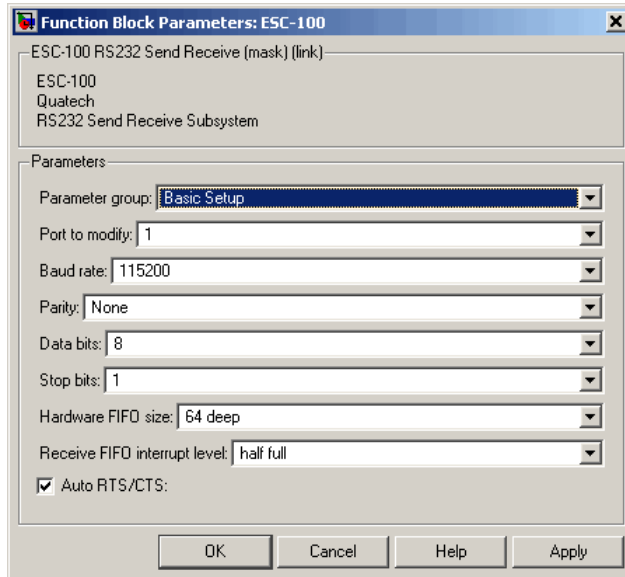
Be sure that these addresses do not conflict with the COM port addresses listed in “Adding RS-232 Blocks” on page 2-6. Note that if you set the configuration register address to 0x2c0, it conflicts with COM2. If you set the configuration address to 0x3c0, it conflicts with COM1.

### **First port address** (Diamond Emerald-MM)

From the list, select the first port address for the board. This address is the first of four port addresses for the Emerald-MM board. You must initially set this address on the board with a jumper. The remaining three addresses follow consecutively in increments of 0x8. Be sure that these addresses do not conflict with the COM port addresses listed in “Adding RS-232 Blocks” on page 2-6.

# RS-232/RS-422/RS-485 Send/Receive

Parameter Group: Basic Setup



## Port to modify

From the list, choose a port. The **Port to modify** parameter specifies the port for which you want to view or modify the parameters.

For Baseboard drivers, the port is the Simulink block port, where the upper port is 1 and the lower port is 2. For all other drivers, the port number corresponds to the channel number.

## Baud Divisor (Commtech)

Enter a divisor integer. The block determines the actual baud rate for a particular channel by dividing the maximum baud rate by this divisor. This number can be different for each channel.

## Baud rate (Mainboard, Quatech, Diamond)

From the list, choose a baud rate.

(Diamond Emerald-MM) The Emerald-MM block lists the baud rates with an HS label to indicate the rate if the board contains the high speed crystal option.

## **Parity**

From the list, choose None, Even, Odd, Mark, or Space. This parameter defines the parity.

## **Data bits**

From the list, choose either 5, 6, 7, or 8 to select the number of bits per character.

## **Stop bits**

From the list, choose either 1 or 2 to define the number of stop bits for the port. Most modern hardware works fine with a character stream that uses single (1) stop bits.

## **Hardware FIFO size** (Quatech, Diamond)

From the list, choose either 64 deep, 16 deep, or 1 deep. This parameter specifies the size of the FIFO in the UART. The **Hardware FIFO size** parameter affects both the receive and transmit FIFOs. For example, specifying a FIFO size of 64 bytes results in fewer interrupts. Fewer interrupts can allow more processing to occur in the model.

The types of UARTs include

16450 — Maximum 1 byte depth

16550 — Maximum 16 byte FIFO depth

16750 — Maximum 64 byte FIFO depth

## **Receive FIFO interrupt level**

From the list, choose 1, quarter full, half full, or almost full. This parameter specifies the number of characters in the Receive FIFO before an interrupt occurs. Receive interrupts occur at least as often as this parameter specifies.

## RS-232/RS-422/RS-485 Send/Receive

---

If a gap of at least 4 character times, the span of four characters, occurs in a data stream, the UART requests an interrupt for the receiver. The UART requests an interrupt regardless of the value of **Receive FIFO interrupt level**. If there is at least one character in the hardware FIFO, an interrupt is signaled.

(Commtech) The Fastcom 422/2-PCI hardware FIFO is fixed at 128 bytes for receive and transmit. By default, this parameter is set to 64. In general, leave this value at 64. If receive latency is critical, you might want to reduce this value to the size of your receive data packet.

### **Auto RTS/CTS** (RS-232 boards)

Select this check box to enable the hardware-based handshake for flow control. This RTS/CTS handshake feature of the UART provides a reliable way to prevent loss due to hardware FIFO overflow.

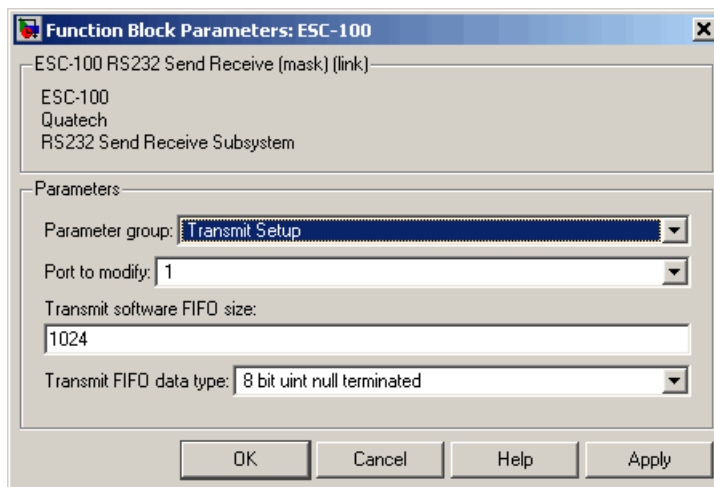
Because of the large 64 byte FIFO in the hardware, flow control that is based on software control in the interrupt service routine can have problems. In most cases, the interrupt service routine executes quickly enough to empty the hardware FIFO. However, if you get hardware FIFO overruns, select this check box. Doing so causes the hardware to stop sending data.

### **Assert on transmit** (RS-422/485 boards)

From the list, select None, RTS, or DTR to specify the state of the RTS or DTR line. The block asserts the selected line upon transmission.

# RS-232/RS-422/RS-485 Send/Receive

Parameter Group: Transmit Setup



## Port to modify

From the list, choose a port. The **Port to modify** parameter specifies the port for which you want to view or modify the parameters.

## Transmit software FIFO size

Enter the transmit software FIFO size, in bytes. This parameter specifies the size of the software FIFO used to buffer transmitted characters.

## Transmit FIFO data type

From the list, choose count+32 bit int, count+32 bit uint, count+16 bit int, count+16 bit uint, 8 bit int null terminated, or 8 bit uint null terminated. This parameter specifies the data type of the transmitter. The 8-bit data types require a NULL-terminated string in the input vector.

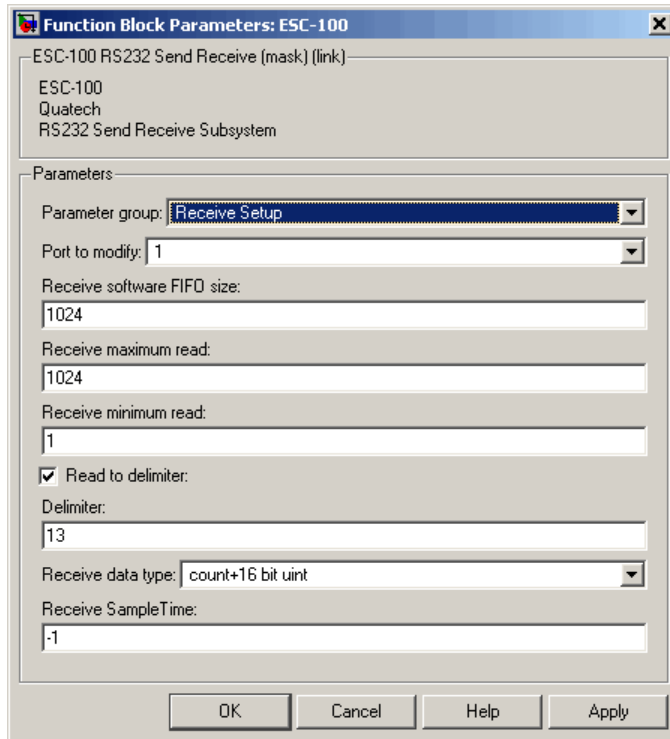
The 16- and 32-bit data types reserve the first full element to contain the number of elements to expect in the rest of the input vector. Only the low-order byte of each data element is sent.

# RS-232/RS-422/RS-485 Send/Receive

Setting this data type allows a wider data type to hold the bytes. If the data stream needs to include a NULL byte, you must select one of the 16- or 32-bit data types.

(Non-F Send Receive blocks)

Parameter Group: Receive Setup



## Port to modify

From the list, choose a port. The **Port to modify** parameter specifies the port for which you want to view or modify the parameters.



## **Receive software FIFO size**

Enter the size of the receive software FIFO, in bytes. This parameter specifies the size of the software FIFO to buffer characters between interrupt service and periodic execution.

## **Receive maximum read**

Enter the maximum number of elements that you want returned by a single call to this block. This parameter is also used to set the output vector width. If the **Read to delimiter** check box is selected, the maximum number of characters read is limited by this parameter even if the delimiter is not found.

## **Receive minimum read**

Enter the minimum number of characters to read. If the FIFO does not contain at least this number of characters, the output vector is empty.

## **Read to delimiter**

Select this check box to have this block return all characters in the FIFO up to and including the specified delimiter. If the block does not find the delimiter in the FIFO, it returns no characters.

Note that if the buffer has hardware observed errors, such as framing errors, characters are returned regardless of the presence of the delimiter. This special case helps diagnose errors such as mismatched baud rates.

## **Delimiter**

Enter the numeric value of the character that is the message delimiter. Any value from 0 to 255 is valid. The common case looks for 10 (line feed) or 13 (carriage return).

## **Receive data type**

From the list, select count+32 bit int, count+32 bit uint, count+16 bit int, count+16 bit uint, 8 bit int null terminated, or 8 bit uint null terminated. This parameter specifies the data type of the receiver. The 8-bit data types produce a null terminated string in the output vector. For 16- and

# RS-232/RS-422/RS-485 Send/Receive

---

32-bit data types, the first element contains the number of valid elements in the rest of the output vector.

For 8-bit data types, only the character data is in the output vector, and a NULL terminator is appended. The 16- or 32-bit wide data types cause the error status from the UART to be placed in the second byte of each data element. (The error status contains the parity, overrun, framing, and break bits.) The character data is in the bottom eight bits of each element; the first element of the vector contains the number of data elements that follow.

## **Receive SampleTime**

Base sample time or a multiple of the base sample time.

(Non-F Send Receive blocks)

Parameter Group: FIFO Setup

(All F Send Receive blocks)

## **Port to modify**

From the list, choose port 1 or 2. The **Port to modify** parameter specifies the port for which you want to view or modify the parameters.

## **Transmit software FIFO size**

Enter the transmit software FIFO size, in bytes. This parameter specifies the size of the software FIFO used to buffer transmitted characters.

## **Transmit FIFO data type**

From the list, choose count+32 bit int, count+32 bit uint, count+16 bit int, count+16 bit uint, 8 bit int null terminated, or 8 bit uint null terminated. This parameter specifies the data type of the transmitter. The 8-bit data types require a NULL-terminated string in the input vector.

The 16- and 32-bit data types reserve the first full element to contain the number of elements to expect in the rest of the input

vector. Only the low-order byte of each data element is sent. Setting this data type allows a wider data type to hold the bytes. If the data stream needs to include a NULL byte, you must select one of the 16- or 32-bit data types.

### **Receive software FIFO size**

Enter the size of the receive software FIFO, in bytes. This parameter specifies the size of the software FIFO to buffer characters between interrupt service and periodic execution.

# RS-232/422/485 Setup (Composite)

---

<b>Purpose</b>	RS-232/422/485 Setup block
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	<p>A setup block is a subsystem block that sets up the interface characteristics for the board.</p> <p>For Quatech boards, this setup block is for one channel or port.</p>
<b>Block Parameters</b>	<p><b>Port</b> (Quatech, Commtech) From the list, choose a port. The <b>Port</b> parameter defines the port this driver block configures.</p> <p><b>Baud Divisor</b> (Commtech) Enter a divisor integer. The block determines the actual baud rate for a particular channel by dividing the maximum baud rate by this divisor. This number can be different for each channel.</p> <p><b>Baud rate</b> (Mainboard, Quatech, Diamond) From the list, choose a baud rate.</p> <p><b>Number of data bits</b> From the list, choose either 5, 6, 7 or 8 to define the number of data bits for the port.</p> <p><b>Number of stop bits</b> From the list, choose either 1 or 2 to define the number of stop bits for the port.</p> <p><b>Parity</b> From the list, choose None, Even, Odd, Mark or Space. This parameter defines the receive and transfer parity.</p> <p><b>Fifo mode</b> (Mainboard, Quatech, Diamond) From the list, choose 64 deep, 16 deep, or 1 deep. This parameter sets the transmit and receive FIFO depth. The UART can operate with a hardware FIFO depth of 1 character (1 deep), 16 characters (16 deep), or 64 characters (64 deep).</p>

# RS-232/422/485 Setup (Composite)

---

## Receive trigger level

From the list, choose 1, quarter full, half full, or almost full. This parameter defines a trigger level for a receive data available interrupt. When the hardware FIFO reaches the level specified in this parameter, the driver asserts the receive data available interrupt.

## Enable auto RTS/CTS

Select this check box to enable hardware-controlled handshaking using the RTS and CTS modem control lines. If this is not checked, no handshaking is done.

## Base Address (Mainboard, ISA, or Diamond)

Enter the base address of the board that you are setting up.

For Diamond-MM-8 boards, this block uses the base address you enter for the Send/Receive block.

## IRQ (Diamond-MM-8)

This field contains the value of the **IRQ number** parameter from the **Parameter Group: Board Setup** of the Send/Receive block.

For Emerald-MM-8, this block setting programs the board IRQ.

## Slot (PCI boards)

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# RS-232/422/485 Write Hardware FIFO (Composite)

---

**Purpose** RS-232/422/485 Write Hardware FIFO block

**Library** xPC Target Library for RS232

**Description** The Write Hardware FIFO block writes the data from the input port (labeled E) to the hardware FIFO in the UART for this port. The following pseudocode most accurately describes the behavior of this FIFO.

```
if (enable is false)
    return
else
{
    if (input data empty)
        disable transmitter buffer empty interrupt
        return
    else
        copy input data to HW FIFO
}
```

## Block Parameters

**Base address** (Mainboard, ISA boards)  
Enter the base address of the UART for which you want to write the hardware FIFO.

For Diamond-MM-8 boards, this block uses the base address you enter for the Send/Receive block.

**Port**  
From the list, choose a port. This is the port to which this block writes data.

**Assert on Transmit** (RS-422/485 boards)  
Select None, RTS, or DTR. The board asserts either no bit, the RTS bit, or the DTR bit in the modem control register upon data transmission.

For half duplex operation, set the jumper on the board to send either RTS or DTR signals to the transmit enable gate. See the

# RS-232/422/485 Write Hardware FIFO (Composite)

---

user's manual documentation specific to your board for further information.

## **PCI Slot (PCI boards)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# RS232 State

---

<b>Purpose</b>	RS232 State block
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	<p>The RS232 State block monitors the board state information that is present in the vector coming out of a receive port on a send/receive block.</p> <p>The input data vector can be one of Int8, UInt8, Int16, or UInt16. If the input vector is Int8 or UInt8, no error status is available and the Boolean outputs are always false. If the input vector is Int16 or UInt16, the upper byte should contain the error status bits from the UART.</p> <p>This block accumulates errors over the whole input vector. An output error state is true if it is true for any byte in the input vector.</p> <p>The FIFO Hardware FIFO block puts the UART status in 16+32 bit data streams. The RS232 State block looks at this status. Note that only the FIFO Read block passes this status information to its output port.</p>
<b>Block Parameters</b>	<p><b>Overrun error output</b> Select this check box to retrieve overrun error output. This output is true if the hardware FIFO in the UART was filled at any time while a character in the input vector was being received.</p> <p><b>Parity error output</b> Select this check box to retrieve parity error output. This output is true if any byte in the input vector has incorrect parity.</p> <p><b>Framing error output</b> Select this check box to retrieve framing error output. This output is true if a framing error occurs on any character in this vector. For example, a framing error might occur if the baud rates between the transmitter and receiver do not match.</p> <p><b>Break interrupt output</b> Select this check box to retrieve break interrupt output. A break interrupt output is not an error, but the UART treats it like an error state. The break condition is detected if the serial line</p>



remains at logic 0 (negative voltage) for more than one character time.

---

**Note** Disconnecting the serial cable does not cause a break with some serial port hardware.

---

## RS-232 Binary Pack (Conventional)

---

<b>Purpose</b>	RS-232 Binary Pack block
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	Same as UDP Pack block. See UDP Pack.

# RS-232 Binary Receive (Conventional)

---

**Purpose** RS-232 Binary Receive block

**Library** xPC Target Library for RS232

**Description** The RS232 Binary Receive Block was designed with generality in mind. To this end, it supports reception of variable-length packets. A packet can be split between two different RS232 Binary Receive blocks, say for a fixed-length header followed by a variable-length body. However, the maximum possible length of a packet has to be specified in the block, and the output from the block is a vector whose width is equal to this maximum length.

The RS232 Binary Receive Block has two input ports:

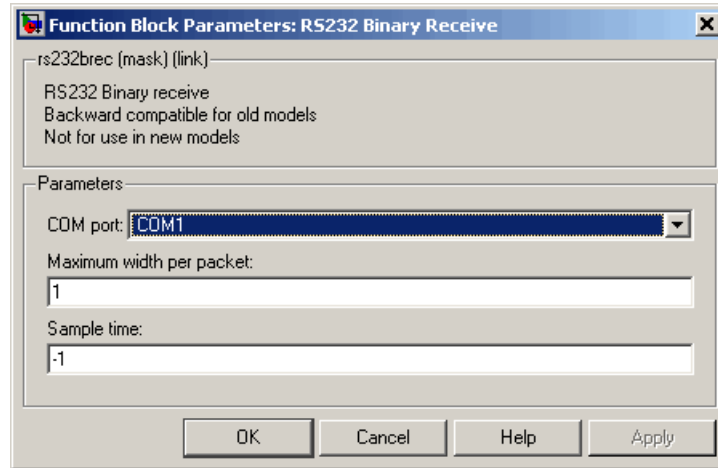
- **First input port** — This port is labeled `Length`, and is the size of the packet it will receive. This value should be less than or equal to the `Maximum width per packet length` parameter. The effect of changing the `Length` input during reception of one packet is undefined.
- **Second input port** — This port is labeled `Enable`, and turns the block on or off. If the `Enable` input is nonzero, the block attempts to receive data, otherwise it simply does nothing.

The RS232 Binary Receive block has two output ports:

- **First output port** — This port is labeled `Done`, and is a function call output. This output issues a function call as soon as the block has completed receiving one packet. This can be used to drive a function call subsystem to switch. For example, to switch from a “header-receive” block to a “body-receive” block.
- **Second output port** — This port is labeled `Data`, and is the data output port. The data is a vector of `uint8s`, and is a vector of width equal to that specified in the `Maximum width per packet` parameter. If the `Length` input is less than this width, the first number of bytes equal to `Length` are the real data and the rest is garbage.

# RS-232 Binary Receive (Conventional)

If you drop a block into your model and double-click it, a Block Parameters dialog box opens where you can modify the parameters for this block.



## Block Parameters

### COM Port

From the list, select COM1, COM2, COM3, or COM4. This is the RS-232 port you want to receive data from. An RS232 Setup block must also exist for the same COM port in your model.

### Maximum width per packet

Enter a value that Simulink and Real-Time Workshop use to allocate memory for the received data. This is also the width of the data output. In case the actual data is less wide than the maximum, the first few bytes of the output vector are the real data and the remaining bytes are undefined.

### Sample time

Specifies how often the block is to be executed. In the example dialog box shown above, the setting of **-1** specifies an inherited sample time, either from the base sample time of the model or from the block that the output of this block goes to.

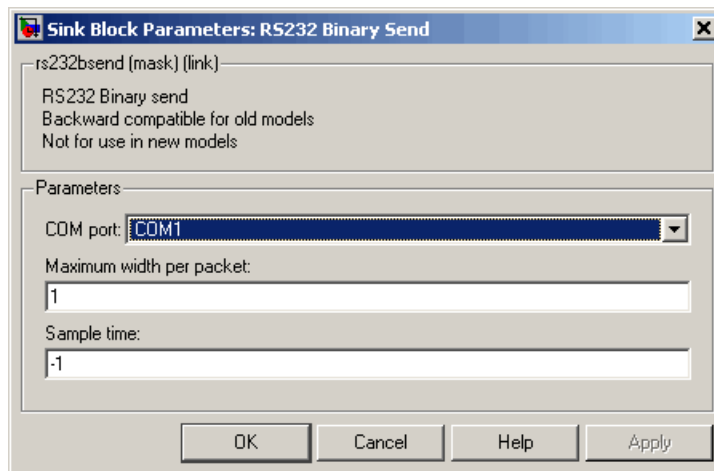
# RS-232 Binary Send (Conventional)

**Purpose** RS-232 Binary Send block

**Library** xPC Target Library for RS232

**Description** If you drop a block into your model and double-click it, a Block Parameters dialog box opens where you can modify the parameters for this block.

This block has one input port. This port represents the data to be transmitted. The data should be a vector of type `uint8` and of a packet width specified in the **Maximum width per packet** parameter.



## Block Parameters

### COM Port

From the list, select COM1, COM2, COM3, or COM4. This is the port you want to use to send the data.

### Maximum width per packet

Enter the width of the incoming data. This value is a constant, unlike the Receive block.

### Sample time

Enter the frequency this data is sent.

## RS-232 Binary Unpack Block (Conventional)

---

<b>Purpose</b>	RS-232 Binary Unpack Block
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	Same as UDP Unpack block. See UDP Unpack.

# RS-232 Mainboard Receive (Asynchronous) (Conventional)

---

<b>Purpose</b>	RS-232 Mainboard Receive block (Asynchronous)
<b>Library</b>	xPC Target Library for RS232
<b>Block Parameters</b>	<p><b>Port</b></p> <p>This list allows you to define which COM port is used to send and receive data. The model must contain one RS232 Setup block for the same COM port. Otherwise, an error message is displayed.</p> <p><b>Message struct name</b></p> <p>Enter the name of the MATLAB structure this block uses to receive messages and data from an RS-232 device. For information on creating this structure, see “Creating RS-232 Message Structures (Asynchronous)” on page 2-39.</p> <p><b>Sample time</b></p> <p>This entry allows you to define the sample time of the block. Because the block does not wait until data is received from the external RS-232 device, you can set sample times to small values.</p>

# RS-232 Mainboard Send (Asynchronous) (Conventional)

---

<b>Purpose</b>	RS-232 Mainboard Send block (Asynchronous)
<b>Library</b>	xPC Target Library for RS232
<b>Block Parameters</b>	<p><b>Port</b></p> <p>This list allows you to define which COM port is used for sending data. The model must contain one RS232 Setup block to configure its COM port. Otherwise, an error message is displayed.</p> <p><b>Message struct name</b></p> <p>Enter the name of the MATLAB structure this block uses to send messages and data to an RS-232 device. For information on creating this structure, see “Creating RS-232 Message Structures (Asynchronous)” on page 2-39.</p> <p><b>Sample time</b></p> <p>This entry allows you to define the sample time of the block. Because the block does not wait until data is received from the external RS-232 device, you can set sample times to small values.</p>



# RS-232 Mainboard Send/Receive (Synchronous) (Conventional)

---

<b>Purpose</b>	RS-232 Mainboard Send/Receive Block (Synchronous)
<b>Library</b>	xPC Target Library for RS232
<b>Block Parameters</b>	<p><b>Port</b></p> <p>From the list, select COM1, COM2, COM3, or COM4. This list allows you to define which COM port is used to send and receive the data. The model must contain one Setup block for each COM port you use to send and receive data. Otherwise, an error message is displayed. Note that data is sent and received on the same COM port.</p> <p><b>Message struct name</b></p> <p>Enter the name of the MATLAB structure this block uses to send and receive messages and data to an RS-232 device. For information to create this structure, see “Creating RS-232 Message Structures (Synchronous)” on page 2-28.</p> <p><b>Sample time</b></p> <p>This entry allows you to define the sample time of the block. Because this block waits for data to be received from the RS-232 external device before the block finishes executing, small sample times are not suitable with synchronous mode. You must allow sufficient time for both the RS232 send and the RS232 receive operations to be completed. The smallest sample time depends on the following:</p> <ul style="list-style-type: none"><li>• Amount of data being sent</li><li>• Amount of data being received</li><li>• Selected baud rate</li><li>• Response time of the external device</li></ul>

# RS-232 Mainboard Setup (Conventional)

---

**Purpose** RS-232 Mainboard Setup block

**Library** xPC Target Library for RS232

**Description** The RS232 Setup block defines the number of databits, baudrate, protocol, and so on for each COM port used in your Simulink model. Each model that uses RS232 I/O must have one RS232 Setup block for each COM port in the model. The RS232 Setup block does not have any inputs or outputs.

If your host PC and target PC are connected using serial communication, one COM port on your target PC is dedicated for communication with your host PC. You cannot use this COM port in your block diagram as an I/O device. For example, if the target PC uses COM1 for communication with the host PC, COM1 cannot be used by your block diagram. An error message is displayed if you use COM1 as an I/O device in your block diagram. The error message appears when you attempt to build and download the target application. In this example, you must use COM2 as an I/O device in your block diagram. If you are using TCP/IP as your host PC to target PC communications protocol, then you can use any COM ports with RS-232 I/O drivers in your block diagram.

## Block Parameters

**Port**  
From the list, select COM1, COM2, COM3, or COM4. This is the serial connection the target PC uses to communicate with the RS-232 device.

**Baud rate**  
From the list, select 115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200, 300, or 110.

**Number of data bits**  
From the list, select either 7 or 8.

**Number of stop bits**  
From the list, select 1 or 2.

**Parity**  
From the list, select None, Odd, or Even.

# RS-232 Mainboard Setup (Conventional)

---

## **Protocol**

From the list, select None or X0nX0ff. If your serial device does not support hardware handshaking, or your application software requires X0n/X0ff handshaking, you might need to select X0n/X0ff.

## **Send buffer size**

Enter the size, in bytes, of the send buffer.

## **Receive buffer size**

Enter the size, in bytes, of the receive buffer.

The **Send Buffer Size** and **Receive Buffer Size** must be large enough to hold the data to be sent or received during each model step. It is important to be aware that the buffers must also be large enough to store old data from a prior model step in the event that the entire data transmission was not completed during the prior step.

## **Initialization command structure**

Enter the name of the structure containing the initialization messages and the expected acknowledgments when the model is initialized. If you are not using initialization messages, enter an empty matrix in this box.

For information on creating this structure, see “Creating RS-232 Message Structures (Synchronous)” on page 2-28 and “Creating RS-232 Message Structures (Asynchronous)” on page 2-39.

## **Termination command structure**

Enter the name of the structure containing the termination messages and expected acknowledgments when the model is terminated. If you are not using termination messages, enter an empty matrix in this box.



# GPIB I/O Support

---

xPC Target interfaces the target PC to a GPIB instrument bus using an external GPIB controller from National Instruments. This external controller is connected to the target PC with a serial cable. This chapter includes the following sections:

Introduction to GPIB Drivers (p. 3-2)	Description of hardware connections, Simulink blocks, and MATLAB message structures associated with the Simulink blocks
Using GPIB Drivers (p. 3-5)	Procedures to add GPIB driver blocks to your Simulink model and create the message structures associated with those blocks
GPIB MATLAB Structure Reference (p. 3-13)	Description of the fields in the message structures, shortcuts, and data types supported in the message fields
Boards and Blocks — Alphabetical List (p. 3-20)	Description of block and parameters for GPIB driver blocks

## Introduction to GPIB Drivers

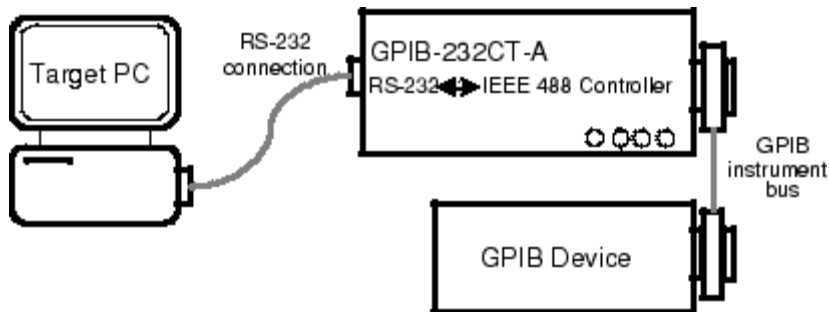
xPC Target uses a model for I/O that includes both Simulink blocks, for the I/O drivers, and MATLAB structures for sequencing messages and commands. This model provides increased flexibility and control over using only Simulink blocks in your model. The topics in this section are

Hardware Connections for GPIB (p. 3-2)	Connect the target PC to a GPIB-232CT-A controller from National Instruments.
Simulink Blocks for GPIB (p. 3-3)	Add setup, send, and receive blocks to your Simulink model.
MATLAB Message Structures for GPIB (p. 3-3)	Create message structures to sequence instructions to and from the GPIB controller.

## Hardware Connections for GPIB

xPC Target supports connecting to a GPIB instrument bus with a GPIB-232CT-A controller from National Instruments.

One end of the controller is connected to either the COM1 or COM2 port on the target PC with a null modem cable. The other end is connected to the GPIB instrument bus with a standard GPIB connector and cable.



## Simulink Blocks for GPIB

To support the use of GPIB, the xPC Target I/O library includes a set of GPIB driver blocks. These driver blocks can be added to your Simulink model to provide inputs and outputs to devices on a GPIB instrument bus.

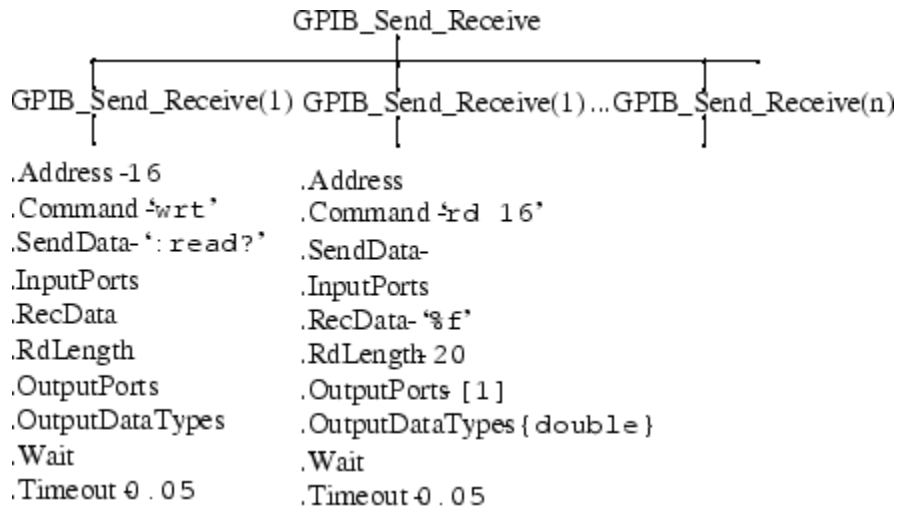
- GPIB Setup — One setup block is needed for each GPIB controller. The setup block does not have any inputs or outputs, but sends the initialization and termination messages.
- GPIB Send/Receive — The send/receive block has inputs and outputs from your Simulink model, and sequences both the send and receive messages.

## MATLAB Message Structures for GPIB

Communication is through a series of messages passed back and forth between the target PC and the GPIB controller. To accomplish this, the messages sent to the GPIB controller must be in a format that the controller understands. Likewise, the target PC must know how to interpret the data returned from the GPIB controller.

xPC Target uses MATLAB structures to create messages and map the input and output ports on the GPIB driver blocks to the data written and read from the GPIB devices. The GPIB Setup block executes the messages in the initialization structure after downloading the target application. The GPIB Send/Receive block repeats the execution of the messages in the send/receive structure during each sample interval. When the target application stops running, the GPIB Setup block executes the messages in the termination structure.

Below is an example of a send/receive message structure. The first message writes a command to instruct the GPIB device to acquire a single data value, while the second message sends a command to read that value and output the result to the output port line coming from a GPIB driver block.



Currently, only two limitations exist. xPC Target does not support string data types as input and output data types. Also, you must know the size and order of data returned from a read command.

For more information on this example, see “Creating GPIB Message Structures” on page 3-9.



## Using GPIB Drivers

xPC Target uses a combination of Simulink blocks and MATLAB structures to support GPIB communication from your target application and target PC. The topics in this section are

Adding GPIB Driver Blocks (p. 3-5)	Add the setup and send/receive blocks you need to add to your Simulink model for GPIB communication.
Creating GPIB Message Structures (p. 3-9)	Create the initialize, send/receive, and termination message structures you need in the MATLAB workspace.

This section uses an example of a multimeter attached to a GPIB bus with an address of 16. This multimeter needs the initialization command

```
:conf:volt:dc
```

to set the device to read DC voltages, and needs the command

```
:read?
```

during each sample interval to read one voltage value

### Adding GPIB Driver Blocks

The GPIB driver blocks initialize and communicate directly with the GPIB controller. The GPIB controller then communicates with the GPIB devices on the instrument bus.

After you create a Simulink model, you can add GPIB driver blocks and define the initialization, send/receive, and termination message structures.

**1** In the MATLAB Command Window, type

```
xpcLib
```

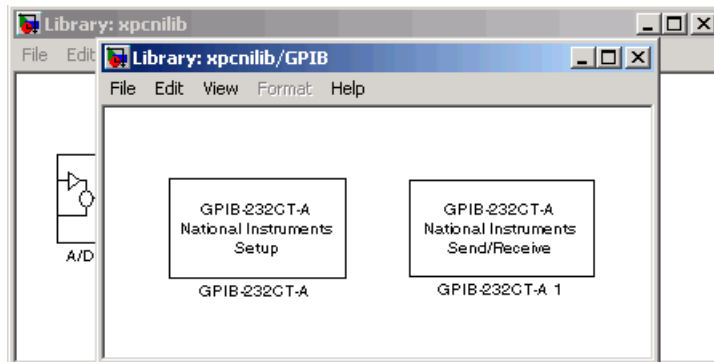
The xPC Target driver block library opens.

- 2 Double-click the GPIB group block.

A manufacturers window opens. Currently xPC target only supports GPIB communication with a National Instruments controller.

- 3 Double-click the National Instruments group block.

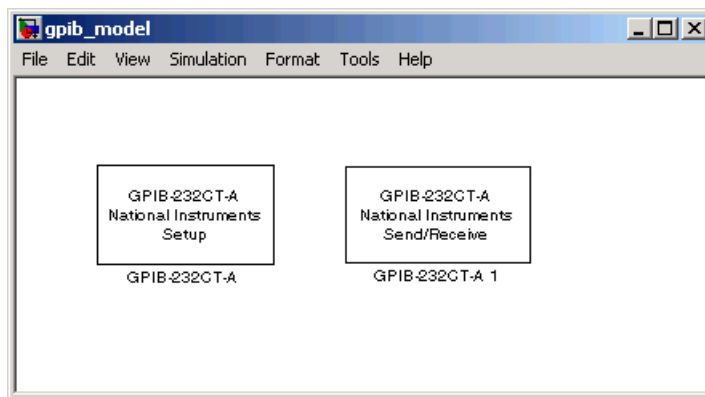
A window with blocks for GPIB drivers opens.



Alternatively, you could access the xPC Target block library from the Simulink Library Browser. In the Simulink window, and from the **View** menu, click **Show Library Browser**. In the left pane, double-click **xPC Target**, double-click **GPIB**, and then click **National Instruments**.

- 4 Drag and drop a GPIB Setup block and a GPIB Send/Receive block to your Simulink model.

Your model should look similar to the figure below. Note that the input and output ports are not defined or visible on the blocks. The inputs and outputs are defined in a MATLAB message structure, and are visible only after you load that structure into the MATLAB workspace and update your Simulink model.



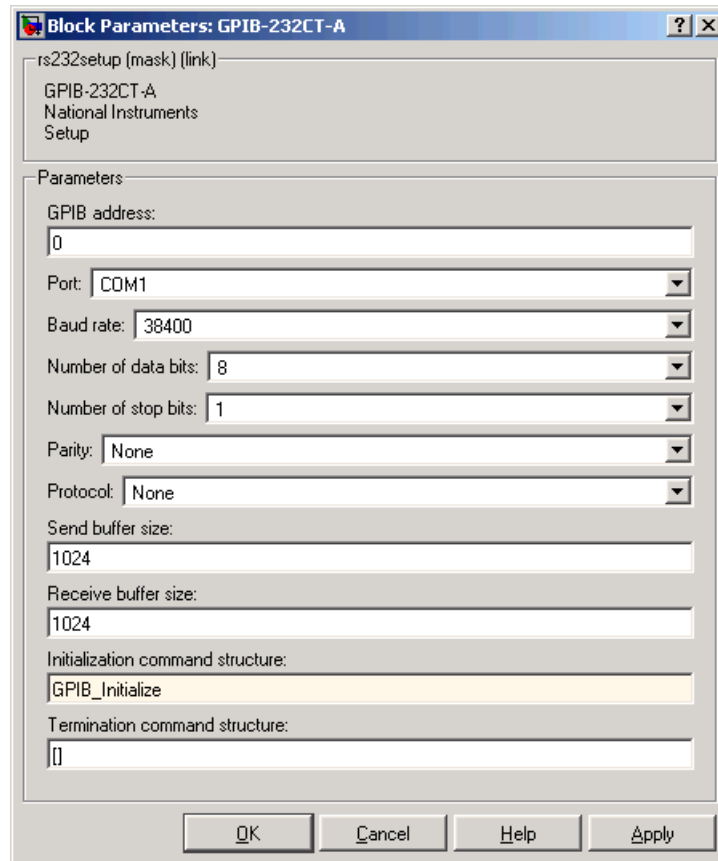
- 5 Double-click the GPIB Setup block. Enter values that correspond to the DIP switch settings you set on the GPIB-232CT-A controller. In the **Initialization Struct** box, enter the name for the MATLAB structure this block uses to send initialization messages to the GPIB device.

---

**Note** If you are not using an initialization or termination structure, enter two single quotes.

---

For example, if the target PC is connected to COM1, and you set the switches on the controller to 38400 baud, 8 data bits, and 1 stop bit, your Block Parameter dialog box should look similar to the figure shown below.

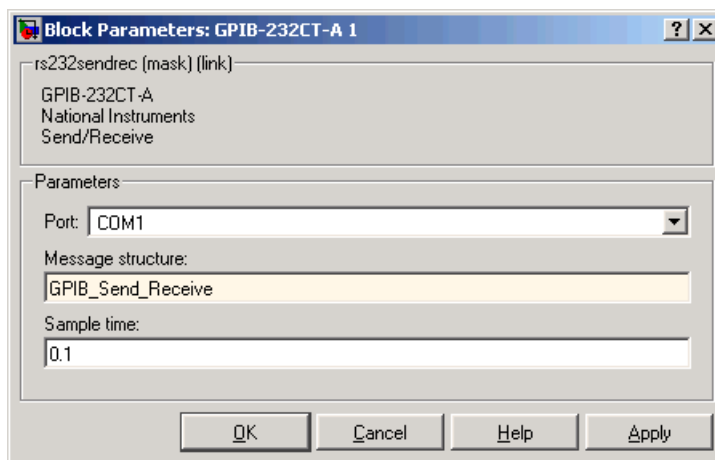


For more information on entering the block parameters, see National Instruments PC-DIO-24. For the procedure to create the initialization structure, see “Creating GPIB Message Structures” on page 3-9.

- 6** Click **OK**. The Block Parameters dialog box closes.
- 7** Double-click the GPIB Send/Receive block. The Block Parameters dialog box opens.

- 8** From the **Port** list, select either COM1 or COM2. This is the port on the target PC connected to the GPIB controller. In the **Message Struct Name** box, enter the name for the MATLAB structure this block uses to send and receive messages to the GPIB device. In the **Sample Time** box, enter the same sample time or multiple of the sample time you entered for the fixed step size in the **Simulation > Configuration Parameters** dialog box **Solver** pane.

Your Block Parameter dialog box should look similar to the figure shown below.



For more information on entering the block parameters, see National Instruments PC-DIO-24.

- 9** Click **OK**. The Block Parameters dialog box closes.

Your next task is to create the MATLAB message structures that the GPIB driver blocks use to sequence commands to the GPIB controller. See “Creating GPIB Message Structures” on page 3-9.

## Creating GPIB Message Structures

GPIB drivers use MATLAB structures to send and receive messages and to map the input and output ports on the GPIB driver blocks to the data written and read from the GPIB devices.

After you add GPIB driver blocks to your Simulink model, you can create the message structures to communicate with the GPIB controller. You need to create and load these structures into the MATLAB workspace before you build your target application. The easiest way to create these structures is to create an M-file and load that M-file into the MATLAB workspace.

- 1** In the MATLAB Command Window, and from the **File** menu, point to **New**, and then click **M-file**.

A MATLAB text editor window opens.

- 2** Enter the initialization and send/receive messages. Each message is an element in a MATLAB structure array with a series of fields. For information and examples of these fields, see “GPIB Initialization and Termination Message Structures” on page 3-13 and “GPIB Send/Receive Message Structure” on page 3-14.

As an example, if you have a multimeter attached to a GPIB bus that has an address of 16, needs the initialization command `:conf:volt:dc` to set the device to read DC voltages, and uses the command `:read?` to read one voltage value, you could type the following:

---

**Note** Field names in the structures are case sensitive.

---

```
GPIB_Initialize(1).Command = 'wrt 16';
GPIB_Initialize(1).SendData = ':conf:volt:dc';

GPIB_Send_Receive(1).Address= 16;
GPIB_Send_Receive(1).Command = 'wrt 16';
GPIB_Send_Receive(1).SendData = ':read?';
GPIB_Send_Receive(1).Timeout = 0.05;

GPIB_Send_Receive(2).Command = 'rd 16';
GPIB_Send_Receive(2).RecData = '%f';
GPIB_Send_Receive(2).RdLength = 20;
GPIB_Send_Receive(2).OutputPorts = [1];
GPIB_Send_Receive(2).OutputDataTypes = {'double'};
GPIB_Send_Receive(2).Timeout = 0.15;
```

This example did not need a termination structure. But if it did, the format of the structure is the same as the initialization structure. For example, a termination structure could have a message with the `.Command` and `.SendData` fields.

```
GPIB_Termination(1).Command
GPIB_Termination(1).SendData
```

- From the **File** menu, click **Save As**. In the Save As File dialog box, enter the name of the M-file. For example, enter

```
GPIB_Messages.m
```

- Close the text editing window.

- In the MATLAB Command Window, type the name of the M-file you created with the GPIB structures. For example, type

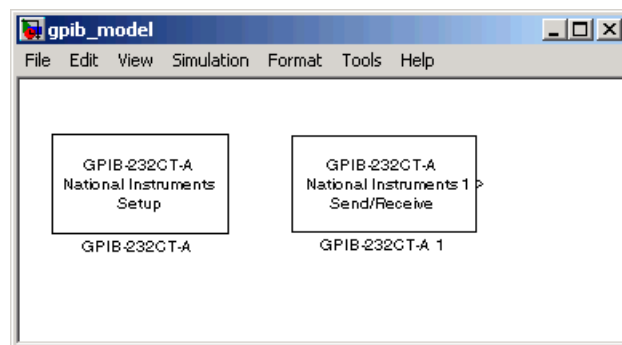
```
GPIB_Messages
```

MATLAB loads and runs the M-file to create the message structures in the MATLAB workspace needed by the GPIB driver blocks.

- Open your Simulink model, or press **Ctrl+D**.

The GPIB driver blocks are updated with the information from the structures. For example, inputs and outputs defined in the structures are now visible on the driver blocks.

Your model should look similar to the figure shown below.



- 7 Set the `PreLoadFcn` for your Simulink model to load the message structures when you open the model. For example, if you saved the message structures in the M-file `GPIB_messages`, type

```
set_param(gcs, 'PreLoadFcn', 'GPIB_messages.m')
```

---

**Note** If you do not manually load the message structures before opening your Simulink model, or have the message structures automatically loaded with the model, the port connections to the GPIB driver blocks break.

---

Your next task is to build the target application and download it to the target PC.



## GPIB MATLAB Structure Reference

You do not use all message fields in all messages. For example, a message to send data would not use the message field `.RecData`, but would use the field `.SendData`. However, knowing the possible message fields is helpful when you are creating any of the message structures.

This section includes the following topics:

GPIB Initialization and Termination Message Structures (p. 3-13)	Description of the message fields for the initialization and termination structures associated with the GPIB Setup block
GPIB Send/Receive Message Structure (p. 3-14)	Description of the message fields for the initialization and termination structures associated with the GPIB Setup block
Shortcuts and Features for Messages (p. 3-17)	Shortcuts to using the GPIB <code>wrt</code> and <code>rd</code> commands
Supported Data Types for Message Fields (p. 3-19)	List of supported data types and the format you use to indicate those types in message fields

### GPIB Initialization and Termination Message Structures

The formats for the initialization and termination structures are similar to the send/receive structure except for a few differences:

The initialization and termination structures do not need to receive or send information through driver block ports on your Simulink model. Therefore, the initialization and termination structures do not use the message fields `.InputPorts`, `.OutputPorts`, `.RecData`, and `.OutputDataTypes`.

Below is a description of the possible message fields for the initialization and termination structures. The order of the message fields does not matter. However, the field names are case sensitive.

<b>Message Fields</b>	<b>Description</b>
Address	Sets the GPIB address for the device being accessed and defines the keyword ADDR. Default value = [ ].
Command	GPIB command sent to a GPIB device. Default value = ''.
SendData	Data sent with the GPIB command. Default value = ''.
RdLength	Defines the length of the acknowledge string, in bytes, from the GPIB controller.
Ack	The expected acknowledgment string from the controller as a result of an initialization or termination message. If this value is set, you need to set the time-out value. If no string is defined, then no acknowledge is expected.
Timeout	Time, in seconds, allowed for the GPIB controller to respond to a message and send back an acknowledge string. Default value = 0.049 seconds.  If the time-out value is exceeded, a time-out error is reported.

### **GPIB Send/Receive Message Structure**

Below is a description of the possible fields for the send/receive message structure. The order of the message fields in a message does not matter. However, the field names are case sensitive.

<b>Message Fields</b>	<b>Description</b>
Address	<p>Sets the GPIB address for the device being accessed. After the GPIB address is set, the remaining messages use this address value until another message changes the address value. Default value = 0.</p> <p>The keyword ADDR is equal to the value in the message field .Address. You can use this keyword in the message fields .Command or .SendData to replace the numerical value of the GPIB address. For example, you can write</p> <pre>GPIB_Send_Receive(1).Command='wrt 16';</pre> <p>Or you can write</p> <pre>GPIB_Send_Receive(1).Address = 16; GPIB_Send_Receive(1).Command='wrt ADDR';</pre>
Command	GPIB command sent to a GPIB device. Default value = ''.
SendData	Data sent with the GPIB command. Default value = ''.
InputPorts	<p>Defines the input ports for the driver block. Data from the input ports is sent to the GPIB device with the message fields .Command and .SendData. Default value = []. The highest number you enter determines the number of input ports on the driver block.</p> <p>For example, the following message creates two input ports on the driver block, and passes data from the input ports to the read command.</p> <pre>GPIB_Send_Receive(1).Command = 'rd %#d %d'; GPIB_Send_Receive(1).InputPorts= [1 2];</pre> <p>The first port is used to dynamically provide the length of the receive string, while the second port provides the value of the GPIB device.</p>

<b>Message Fields</b>	<b>Description</b>
RecData	<p>Format of the data received from the GPIB device. Default value = ''. The format of this statement is very similar to a scanf statement. The read data is mapped to the output ports defined in the field .OutputPorts. If a negative output port is given, the data is read in, but not sent to any output port.</p> <p>For example, to read from a GPIB device with an address of 16, one floating-point number with a maximum number of bytes of 20, and send the data to the first driver block output, type the following:</p> <pre>GPIB_Send_Receive(1).Command = 'rd #20 16'; GPIB_Send_Receive(1).RecData = '%f'; GPIB_Send_Receive(1).OutputPorts = [1];</pre>
RdLength	<p>Defines the length of the data, in bytes, received with the read command and defines the keyword LENGTH. Default value = 0.</p>
OutputPorts	<p>Defines the output ports from the driver block. Data received from a GPIB device with the read command is sent to the output ports. Default value = []. The highest number you enter determines the number of output ports on the driver block.</p> <p>For example, to use output ports 1 and 2 on the driver block, type</p> <pre>GPIB_Send_Receive.OutputPorts = [1 2];</pre>
OutputData Types	<p>Defines the data types for the output ports on the driver block. Default value = [].</p> <p>If this value is not defined, and there are output ports, the default type is double. Also, if there are more output ports than output data types listed, the default type for the undefined ports is double.</p>

Message Fields	Description
Wait	The amount of time, in seconds, to wait before executing the next message. This value is limited to 50 milliseconds. Default value = 0.
Timeout	Time, in seconds, the driver block waits for data to be returned. Default value = 0.049.

### Shortcuts and Features for Messages

xPC Target defines the abbreviations `wrt` and `rd` to make message writing with GPIB commands easier. When the message interpreter sees the statements

- `Structure_name(index) . 'wrt'`, it is replaced with `Structure_name(index) . 'wrt ADDR'`. For example, you could write

```
GPIB_Initialize(1).Command = 'wrt 8';
```

or you could write

```
GPIB_Initialize(1).Address = 8;
GPIB_Initialize(1).Command = 'wrt';
```

The following message fields, with the keyword `ADDR`, use the address value 8 defined in the message field `.Address`.

- `Structure_name (index).Command = 'rd'`, it is replaced with `Structure_name(index).Command = 'rd #LENGTH ADDR'`. For example, you could write

```
GPIB_Initialize(1).Command = 'rd #10 8';
```

or you could write

```
GPIB_Initialize(1).Address = 8;
GPIB_Initialize(1).RdLength = 10;
GPIB_Initialize(1).Command = 'wrt';
```

If you enter numerical values in the `wrt` and `rd` commands, then the command uses those values instead of the values in the variables `ADDR` and `LENGH`. For example, the following message uses the GPIB address 10 even though the value for `ADDR` is defined as 8.

```
GPIB_Initialize(1).Address = 8;
GPIB_Initialize(1).Command = 'wrt 10';
```

### **Changes to the Read Command**

When a GPIB `rd` command is sent to the GPIB controller, the controller responds with the data and length of data. To make using this command easier, the xPC Target driver block discards the length of data information. For example, using the normal GPIB `rd` command, you could write

```
GPIB_Message(1).Command = 'rd #20 16';
GPIB_Message(1).RecData = '%f%d';
GPIB_Message(1).OutputPorts = [1 -1];
```

The code `%d` reads the length of data and the `-1` discards the length. Using the modified xPC Target `rd` command, you would write

```
GPIB_message(1).Command = 'rd #20 16';
GPIB_message(1).RecData = '%f';
GPIB_message(1).OutputPorts = [1];
```

### **Automatic Addition of Escape Characters**

The message interpreter automatically places the correct escape characters at the end of the message fields `.Command`, `.SendData`, and `.Ack`. However, if you add the escape characters, then the message interpreter does not add additional characters.

The escape characters are `\\`, `\a`, `\b`, `\f`, `\r`, `\t`, `\v`, `\'`, `\"`, and `\n`.

For example, you can write

```
GPIB_Message.Command = 'wrt 16\n';
GPIB_Message.SendData = ':conf:volt:dc\r';
GPIB_Message.Ack = '10\n\r';
```

or you can write the following, and the appropriate escape characters are added.

```
GPIB_Message.Command = 'wrt 16';
GPIB_Message.SendData = ':conf:volt:dc';
GPIB_Message.Ack = '10';
```

## Supported Data Types for Message Fields

The following table lists the supported data types for the message fields .SendData and .RecData:

Format	Description
%c and %C	Single character and wide character
%d or %I	Signed decimal integer
%u	Unsigned decimal integer
%o	Unsigned octal integer
%x or %X	Unsigned hexadecimal integer using 'abcdef' or 'ABCDEF' for the hexadecimal digits.
%e or %E	Exponential format using e or E
%f	Floating point
%g	Signed value printed in f or e format depending on which is smaller
%G	Signed value printed in f or E format depending on which is smaller

## **Boards and Blocks – Alphabetical List**



**Board** GPIB Simulink block

**General Description**

The GPIB-232CT-A is a GPIB controller external to the target PC. It is connected to the target PC with an RS-232 cable.

xPC Target supports this controller with two driver blocks:

- GPIB-232CT-A Send/Receive Block
- GPIB-232CT-A Setup Block

**Board Characteristics**

Board name	GPIB-232CT-A
Manufacturer	National Instruments
Bus type	N/A
Access method	RS232
Multiple block instance support	No
Multiple board support	Yes

# GPIB-232CT-A Send/Receive Block

---

<b>Purpose</b>	GPIB-232CT-A Send/Receive Block
<b>Library</b>	xPC Target Library for GPIB
<b>Block Parameters</b>	<p><b>Port</b></p> <p>From the list, select COM1, COM2, COM3, or COM4. This is the serial connection the target PC uses to send and receive data with the GPIB-232CT-A controller.</p> <p><b>Message structure</b></p> <p>Enter the name of the MATLAB structure containing the messages to be sent to the GPIB controller.</p> <p><b>Sample time</b></p> <p>Enter the base sample time or a multiple of the base sample time you entered for the fixed step size in the <b>Simulation &gt; Configuration Parameters</b> dialog box <b>Solver</b> pane.</p>

<b>Purpose</b>	GPIB-232CT-A Setup block
<b>Library</b>	xPC Target Library for GPIB
<b>Note</b>	The setup block parameters must be set to match the jumper settings on the GPIB-232CT-A controller.
<b>Block Parameters</b>	<p><b>GPIB address</b> Enter the identification number for the GPIB controller. When the GPIB-232CT-A is turned on, the identification number is set to 0.</p> <p><b>Port</b> From the list, select COM1, COM2, COM3, or COM4. This is the serial connection the target PC uses to communicate with the GPIB-232CT-A controller.</p> <p><b>Baud rate</b> From the list, select 115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200, 600, or 300.</p> <p><b>Number of data bits</b> From the list, select 8 or 7.</p> <p><b>Number of stop bits</b> From the list, select 1 or 2.</p> <p><b>Parity</b> From the list, select None, Odd, or Even.</p> <p><b>Protocol</b> From the list, select None or XOn/XOff. If your serial device does not support hardware handshaking, or your application software requires XOn/XOff handshaking, you might need to select XOn/XOff.</p> <p><b>Send buffer size</b> Enter the buffer size in bytes.</p> <p><b>Receive buffer size</b> Enter the buffer size in bytes.</p>

# GPIB-232CT-A Setup Block

---

## **Initialization command struct**

Enter the name of the structure containing the initialization information. For example, enter

GPIB\_Initialize

If you are not using initialization messages, enter two single quotes in this box. For information on creating this structure, see “Creating GPIB Message Structures” on page 3-9.

## **Termination command struct**

Enter the name of the structure containing the termination information.

# CAN I/O Support

---

This chapter includes the following sections:

Introduction (p. 4-3)

xPC Target offers support to connect a target PC to a CAN network using the CAN driver blocks provided by the xPC Target I/O block library. This support is for I/O device drivers for the CAN-AC2-ISA and CAN-AC2-PCI boards from Softing GmbH (Germany).

Model Execution Driven by CAN Messages (Interrupt Capability of CAN Receive Blocks) (p. 4-7)

In certain applications, the model (target application) execution is driven by the pace of an incoming CAN message.

Defining Initialization and Termination CAN Messages (p. 4-11)

The CAN Setup driver blocks for all supported CAN boards allow the definition of CAN messages to be sent out during initialization and termination of the target application.

CAN-AC2 and CANopen Devices (p. 4-14)

xPC Target CAN-AC2 supports CAN specification 2.0a and 2.0b but this does not generally include the CANopen protocol on driver level. Nevertheless it is possible to access CANopen devices by the CAN-AC2 drivers in a general way.

Constructing and Extracting CAN Data Frames (p. 4-15)

CAN data frames have a maximum size of 8 bytes (64 bits). For the CAN driver blocks found in the xPC Target I/O block library, Simulink signals of data type double are used to propagate data frames as an entity.

Detecting Time-Outs When Receiving CAN Messages (p. 4-16)

The Receive driver blocks for all CAN boards allow you to output the timestamp at which the latest corresponding CAN message has been received.

CAN Blocks for the CAN-AC2 (ISA) with Philips PCA 82C200 CAN Controller (p. 4-17)

The driver blocks described here support the CAN-AC2 (ISA) without piggyback modules.

CAN Blocks for the CAN-AC2 (ISA) with Intel 82527 CAN Controller (p. 4-26)

The driver blocks described here support the CAN-AC2 (ISA) with piggyback modules.

CAN Blocks for the CAN-AC2-PCI with Philips SJA1000 CAN Controller (p. 4-36)

The driver blocks described here support the CAN-AC2-PCI.

CAN Blocks for the CAN-AC2-104 (PC/104) with Philips SJA1000 CAN Controller (p. 4-46)

The driver blocks described here support the CAN-AC2-104 (PC/104).

## Introduction

xPC Target offers support to connect a target PC to a CAN network using the CAN driver blocks provided by the xPC Target I/O block library. This support is for I/O device drivers for the CAN-AC2-ISA and CAN-AC2-PCI boards from Softing GmbH (Germany). The CAN driver library allows xPC Target applications to connect to any CAN field bus network for I/O communication or real-time target-to-target communication. The topics in this section are

- “xPC Target CAN Library” on page 4-3
- “CAN-AC2” on page 4-5
- “CAN-AC2-PCI” on page 4-5
- “CAN-AC2-104” on page 4-6

### xPC Target CAN Library

The drivers support CAN specifications 2.0A and 2.0B and use the dynamic object mode of the CAN-AC2 firmware to achieve maximum real-time performance.

The library supports the following CAN boards from Softing GmbH, Germany.

Board Name	Form Factor	Identifier Range	Multiple Board Support
CAN-AC2	ISA	Standard (& Extended with piggyback module)	No
CAN-AC2-PCI	PCI	Standard & Extended	Yes (up to 3)
CAN-AC2-104	PC/104	Standard & Extended	Yes (up to 3)

For more information on the board specifications, visit <http://www.softing.com>.

The xPC Target CAN library intentionally restricts its support to Softing boards with two CAN ports (boards with one channel would be available as well). This is because the two-port versions allow you to check the correct functioning of the board and drivers by just connecting the first CAN port to the second CAN port. This forms a loop-back without you needing to connect

the board to a “real” CAN-network. The `xpcdemos` directory contains simple loop-back test models to test the ISA, PCI, and PC/104 boards. Type the following commands to open the corresponding test models.

<b>Model Name (Command)</b>	<b>Board</b>
<code>xpccanisa</code>	CAN-AC2
<code>xpccanpci</code>	CAN-AC2-PCI
<code>xpccanpc104</code>	CAN-AC2-104

The size of the driver code of the CAN boards supported by the xPC Target block library is significant, and because not all xPC Target applications will use CAN, the CAN library code is not linked by default when building a target application. This makes target applications smaller if no CAN communication functionality is needed. If the model to be built contains CAN driver blocks, xPC Target links in the appropriate CAN library code when necessary.

For each CAN board three driver blocks are provided:

- A setup block, which defines the type of physical connection (baud rate and so forth). Exactly one instance of the setup block must be defined in a model for each physically installed CAN board.
- A send block, which transmits (sends) the data entering the block’s input ports to the connected CAN network. One or more instances of the Send block can be used in a model.
- A receive block, which retrieves (reads) CAN messages received by the board and outputs the data at the corresponding output ports. One or more instances of the Receive block can be used in a model.

The maximum size of the data frame of a CAN message is 8 bytes. This is the same size as the C data type `double` uses on PC-compatible systems. At the same time, the `double` data type is the default data type used for Simulink signals. Therefore the CAN data frame within a Simulink model can be easily represented by a scalar Simulink signal even if the data frame has nothing in common with a double floating point value. The xPC Target CAN library provides a Utility sublibrary that offers bit-packing and bit-unpacking blocks. These blocks are used to pack data types other than doubles into 64 bits (8 bytes or a double) as well as for the opposite operation. This is discussed in



greater detail below. What is important for now is that CAN data frames are represented by Simulink signals of data type double.

All drivers for the supported CAN boards program the boards for the so-called *dynamic object mode*. This is one of three modes the CAN board firmware from Softing can operate in. For a more detailed discussion of the three modes see the board's user manual. Dynamic object mode is best suited for real-time environments where each component of the application must have deterministic time behavior. This is the case for xPC Target, and that is the main reason why this mode has been chosen over the other two modes, which are FIFO and static object mode.

The following paragraphs summarize the differences among the three supported Softing boards.

## **CAN-AC2**

This is the CAN board for the ISA bus offering two CAN ports (high-speed). In its standard hardware configuration it uses the Philips PCA 82C200 CAN controller, which supports standard identifiers only. Piggyback modules are available (one for each port) that replace the Philips CAN controllers with Intel 82527 CAN controllers. The Intel controllers support both standard and extended identifiers. The board is a memory-mapped device and uses a 16 KB address range between 640 KB and 1 MB. We do not recommend this board for new projects; use the CAN-AC2-PCI which is described below instead. Softing plans that no new firmware versions are planned for this board.

## **CAN-AC2-PCI**

This is the CAN board for the PCI bus offering two CAN ports. The CAN controllers used on the board are the SJA1000 from Philips. In its standard hardware configuration the board is designed for both standard and extended identifiers for high-speed CAN. Piggyback modules are available (one for each port) that add low-speed CAN support to switch between high-speed and low-speed CAN. The board is a memory mapped PCI device that uses 64 KB of address space. The address space is assigned automatically by the PCI BIOS of the target PC and lies usually in the range between 2 GB and 4 GB. Any new projects where a desktop PC is used as the target system should use this board and not the ISA board described above.

### **CAN-AC2-104**

This is the CAN board for the PC/104 bus offering two CAN ports. The CAN controllers used on the board are the SJA1000 from Philips. The board offers both standard and extended identifiers for high-speed CAN. A low-speed CAN hardware extension is not available. The board is both I/O mapped and memory mapped. The I/O-mapped area uses a 3 B address range and the memory-mapped area uses a 4 KB address range between 640 KB and 1 MB.

## Model Execution Driven by CAN Messages (Interrupt Capability of CAN Receive Blocks)

In certain applications, the model (target application) execution is driven by the pace of an incoming CAN message. The standard behavior of the xPC Target kernel is to drive the model (target application) in time monotonic fashion (time interrupt). However, the driving interrupt can be replaced by any other hardware interrupt. Because the three supported CAN boards permit the firing of a hardware interrupt upon reception of a specific CAN message, you can replace the timer interrupt line in the kernel by the interrupt line assigned to a CAN board. This leads to a CAN-message-driven execution of the target application.

To set this up, two independent steps are necessary:

- 1 Replace the timer interrupt line in the kernel setup with the board's hardware interrupt line.
- 2 Properly set up the CAN Setup and CAN Receive blocks.

Both steps are slightly different for each of the three supported CAN boards. Therefore the two steps are explained for each board type below.

### CAN-AC2 (ISA)

The CAN-AC2 is an ISA board, and the hardware interrupt line is set by means of hardware jumpers on the board. Refer to the Softing user manual for the board on how to set a certain interrupt line. Select an interrupt line that is not used by any other hardware device in the xPC Target system (for example by the Ethernet card).

- 1 In the Simulink window, and from the **Tools** menu, point to **Real-Time Workshop**, and then click **Options**.

The Configuration Parameters dialog box is displayed.

- 2 Click the **Real-Time Workshop** node.
- 3 Ensure that in the **Target selection** section, the **System target file** field is set to `xpctarget.tlc`.

- 4 Select the **xPC Target options** node.
- 5 In the **Real-time interrupt source** field, select the interrupt line number that you have set using the jumpers on the board.
- 6 Click **OK** and save the model.
- 7 Open the dialog box of the CAN Receive block in the model that defines the CAN message (identifier) to be used to fire the interrupt. Select the **Generate interrupts** check box. Selecting this box declares all CAN messages defined in this Receive block instance through their identifiers as messages that fire an interrupt. In other words, it is not possible to define a single CAN message within the set of defined identifiers to be the only one to fire an interrupt. In most cases only the reception of one specific message is used to drive the application execution. Therefore use at least two instances of the Receive block. One to receive the CAN message that drives the execution (Generate Interrupts selected) and the other for all other normal CAN messages to be received (Generate Interrupts cleared).

### CAN-AC2-PCI

The CAN-AC2 is a PCI board, and the hardware interrupt line is automatically assigned by the PCI BIOS during the initialization of the target system. Use the xPC Target function `getxpcpci` (see `help getxpcpci`) at the MATLAB command prompt to query the target system for installed PCI devices and the assigned resources. Write down the interrupt line number assigned to the CAN-AC2-PCI board.

- 1 In the Simulink window, and from the **Tools** menu, point to **Real-Time Workshop**, and then click **Options**.  
  
The Configuration Parameters dialog box is displayed.
- 2 Click the **Real-Time Workshop** node.
- 3 Ensure that in the **Target selection** section, the **System target file** field is set to `xpctarget.tlc`.
- 4 Select the **xPC Target options** node.
- 5 In the **Real-time interrupt source** field, select the interrupt line number that you retrieved with the `getxpcpci` command.

- 6 Click **OK** and save the model.
- 7 Open the dialog box of the CAN Receive block in the model that defines the CAN message (identifier) to be used to fire the interrupt. Select the **Generate interrupts** check box. Selecting this box declares all CAN messages defined in this Receive block instance through their identifiers as messages that fire an interrupt. In other words, it is not possible to define a single CAN message within the set of defined identifiers to be the only one to fire an interrupt. In most cases only the reception of one specific message is used to drive the application execution. Therefore use at least two instances of the Receive block. One to receive the CAN message that drives the execution (Generate Interrupts selected) and the other for all other normal CAN messages to be received (Generate Interrupts cleared).

## **CAN-AC2-104 (PC/104)**

The CAN-AC2-104 is an ISA board (PC/104), and the hardware interrupt line is set by means of a software setting within the CAN Setup driver block. Note a free interrupt line that is not used by any other hardware device in the xPC target system (for example by the Ethernet card).

- 1 In the Simulink window, and from the **Tools** menu, point to **Real-Time Workshop**, and then click **Options**.

The Configuration Parameters dialog box is displayed.

- 2 Click the **Real-Time Workshop** node.
- 3 Ensure that in the **Target selection** section, the **System target file** field is set to `xpctarget.tlc`.
- 4 Select the **xPC Target options** node.
- 5 In the **Real-time interrupt source** field, select the free interrupt line number that you chose.
- 6 Click **OK** and save the model.
- 7 In the model open the dialog box of the CAN Setup block for the CAN-AC2-104 board. Select the chosen interrupt line in the **Interrupt Line** pop-up menu and close the dialog box. Open the dialog box of the CAN Receive block in the model that defines the CAN message (identifier)

to be used to fire the interrupt. Select the **Generate interrupts** check box. Selecting this box declares all CAN messages defined in this Receive block instance through their identifiers as messages that fire an interrupt. In other words, it is not possible to define a single CAN message within the set of defined identifiers to be the only one to fire an interrupt. In most cases only the reception of one specific message is used to drive the application execution. Therefore use at least two instances of the Receive block. One to receive the CAN message that drives the execution (Generate Interrupts selected) and the other for all other normal CAN messages to be received (Generate Interrupts cleared).

After you complete these two steps, you are ready to build the model. After the downloading has succeeded and the target application execution has been started, the execution is now driven by the selected CAN messages. The execution time information displayed on the target screen is now directly dependent on the reception of the corresponding message. If no message is received the time does not advance. You should ensure that the corresponding CAN message on the other CAN node is only generated if the xPC target application is running, otherwise unexpected interrupt messages might be displayed on the target screen.

## Defining Initialization and Termination CAN Messages

The CAN Setup driver blocks for all supported CAN boards allow the definition of CAN messages to be sent during initialization and termination of the target application (once at the beginning of each application run and once before an application run is stopped). The main purpose for sending these messages is to initialize or terminate other CAN nodes on the network. This is the case, for example, for CANOpen or DeviceNet nodes. Even if xPC Target does not provide direct support of those CAN application layers, communication with those nodes can usually be done over standard CAN messages as long as the nodes have been properly initialized. The initialization and termination fields of the Setup blocks are intended for this purpose.

You define the initialization and termination CAN messages using MATLAB struct arrays with CAN specific field names. This is the same concept as used for the RS-232, GPIB, and general Counter driver blocks found in the xPC Target I/O library. Refer to those driver blocks and their help for additional information about this basic concept.

The CAN Setup block-specific field names are the following:

**Port** — Selects the CAN port over which the message is sent. Valid values are either 1 or 2 (double).

**Type** — Defines whether the message to be sent is of type standard or extended. Valid values are either 'Standard' or 'Extended' (strings).

**Identifier** — Defines the identifier of the message. The value (scalar) itself must be in the corresponding identifier range (standard or extended).

**Data** — Defines the data frame to be sent out along with the CAN message. The value must be a row vector of type double with a maximum length of 8. Each element of the vector defines one byte, where the first element defines the data for byte 0 and the eighth element the data for byte 7. Each element can have a value between 0 and 255 (decimal). The data frame size is defined by the length of the row vector.

**Pause** — Defines the amount of time in seconds the Setup block waits after this message has been sent and before the next message defined in the struct array is parsed and sent. Valid values are between 0 and 0.05 seconds. Some

CAN nodes need some time to settle before they can accept the next message, especially when the message just received puts the node in a new operational mode. Use this field to define those necessary idle times.

### Example

Consider an A/D converter module with a CANOpen interface. After the node is powered up, the module is in preoperational mode, which is common for CANOpen nodes. At least two initialization messages must be sent to the node to make the module fully operational.

The first message puts the node from preoperational into operational mode. The second message programs the module so that each time the converted A/D value differs by more than 10 mV from the former conversion, a CAN message is automatically sent, with the converted value as the data frame.

After the target application starts and the node is properly initialized, the node automatically sends a CAN message, which the xPC target application receives and then processes.

Before the target application execution is actually stopped, the module (node) must be returned to preoperational mode. You do this by sending out one corresponding termination message.

The initialization and termination message struct for this example could look as follows:

```
% put node into operational mode
init(1).port=1;
init(1).type='Standard';
init(1).identifier=1536+11;
init(1).data=[hex2dec('22'),hex2dec('23'),hex2dec('64'),hex2dec ...
('00'),hex2dec('01')];
init(1).pause=0.02;

% program node to send CAN messages with converted A/D values
automatically
init(2).port=1;
init(2).type='Standard';
init(2).identifier=0;
```



```
init(2).data=[hex2dec('01'),11];  
init(2).pause=0;  
  
% put node back into preoperational mode  
term(1).port=1;  
term(1).type='Standard';
```

### **CAN-AC2 and CANopen Devices**

xPC Target CAN-AC2 supports CAN specification 2.0a and 2.0b but this does not generally include the CANopen protocol on driver level. Nevertheless it is possible to access some CANopen devices by the CAN-AC2 drivers in a general way.

CANopen knows two types of messages, i.e., SDO and PDO. SDOs are used to set up or initialize a CANopen device for a certain behavior. PDOs are messages that contain real-time data (i.e., converted A/D values from a analog input device) and are CAN-type messages with no CANopen object, index, and subindex information.

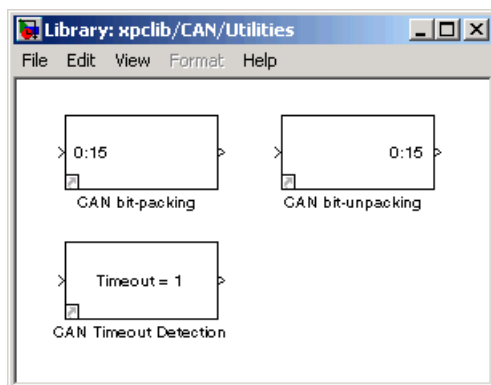
xPC Target applications that have to access CANopen devices over the CAN-AC2 drivers transmit SDOs during the initialization phase and the termination phase of the driver. PDOs are sent or received during the simulation phase of the driver.

Because SDOs and PDOs are regular CAN-messages the CAN-AC2 drivers have to provide a way to transmit SDOs during the initialization and termination phase of the CAN-AC2 Setup block to initialize the different CANopen devices in the network. To do this, provide proper Initialization and termination command structures in the CAN-AC2 Setup block to describe the SDO messages to send to set up and terminate the CANopen device.

## Constructing and Extracting CAN Data Frames

CAN data frames have a maximum size of 8 bytes (64 bits). For the CAN driver blocks found in the xPC Target I/O block library, Simulink signals of data type double are used to propagate data frames as an entity. But in most applications the data frame content does not consist of 64-bit floating point values; instead they are constructed from one or more smaller data type entities such as signed and unsigned integers of various size.

To simplify the construction and extraction of data frames for the user, the xPC Target I/O library contains two utility blocks (found in subgroup CAN/Utilities) that allow bit-packing (construction) and bit-unpacking (extraction) of data frames in a very flexible way.



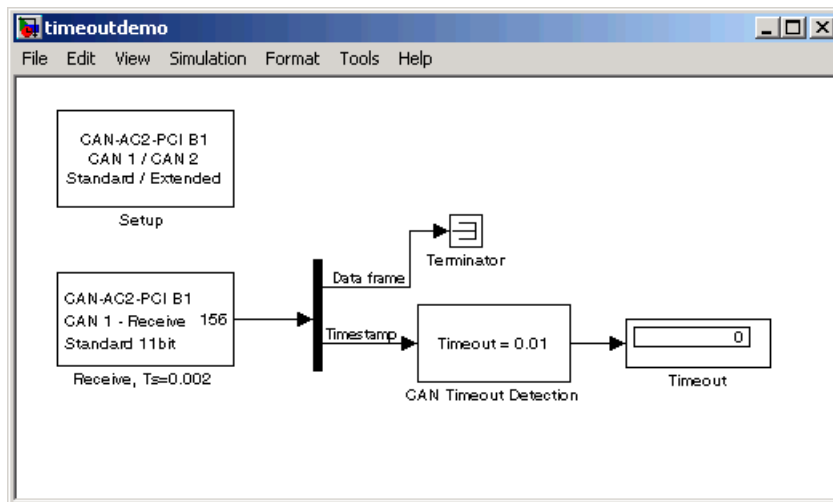
The main purpose of the two blocks is to be used in conjunction with CAN Send and Receive driver blocks, but they can be used as well for other types of data manipulation. Their functionality is entirely independent of any CAN driver blocks or CAN library.

## Detecting Time-Outs When Receiving CAN Messages

The Receive driver blocks for all CAN boards allow you to output the timestamp at which the latest corresponding CAN message was received. This information can be used to detect whether another CAN node is still active and therefore is sending CAN messages or is no longer active and special action must be taken. Assume that a CAN message is expected from another CAN node every 2 milliseconds. If no new message is received within 10 milliseconds, the other CAN node is considered faulty, and the Simulink model (target application) must proceed accordingly.

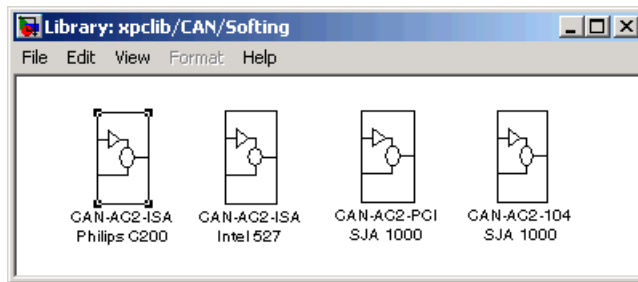
The CAN blockset in the xPC Target I/O block library provides a utility block called CAN Timeout Detection. This is a simple graphical subsystem (inspect it by looking under its mask) that uses the timestamp information to calculate the time-out condition.

A Simulink model using this block in conjunction with a Receive block could look as follows:

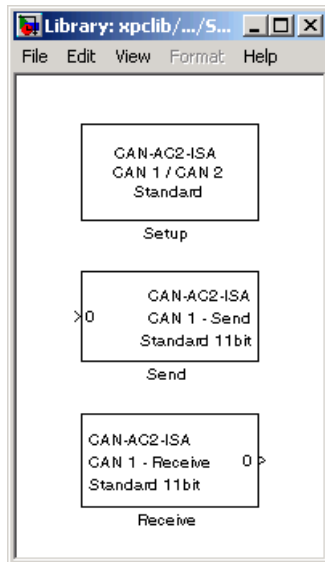


## CAN Blocks for the CAN-AC2 (ISA) with Philips PCA 82C200 CAN Controller

The driver blocks described here support the CAN-AC2 (ISA) without piggyback modules. The Philips PCA 82C200 chip is used as the CAN controller in this configuration and supports the standard identifier range only. The driver block set for this board is found in the xPC Target I/O block library in the group CAN/Softing.



The first group, Phillips C200, contains the three available CAN blocks: Setup, Send, and Receive.

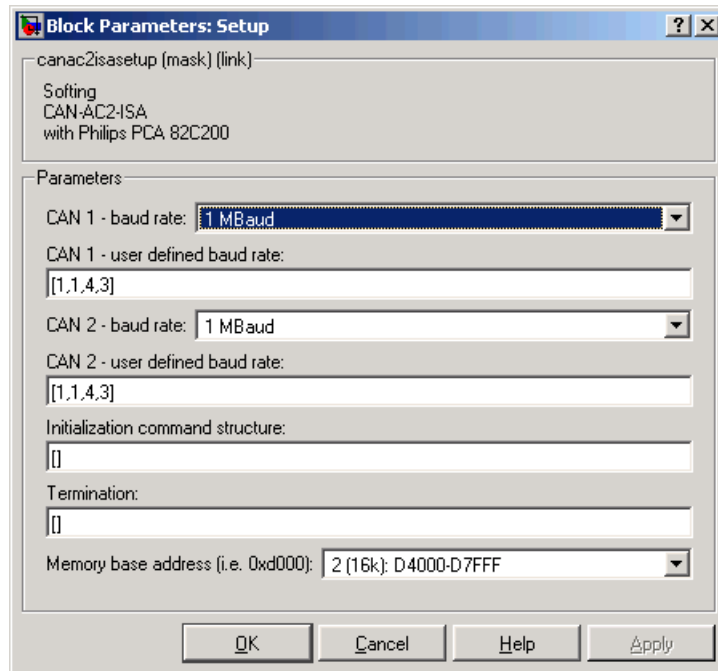


# Softing CAN-AC2-ISA with Philips PCA 82C200 Setup

**Purpose** Softing CAN-AC2-ISA with Philips PCA 82C200 Setup block

**Library** xPC Target Library for CAN

**Description** The Setup block defines general settings of the installed CAN board. Because the CAN driver blocks for this ISA board only support a single physical board for each target system, this block can only be used once (one instance) in a model.



## Block Parameters

### CAN 1 – baud rate

Defines the most common baud rates for CAN port 1. If special timing is necessary (baud rate), select the value User defined. In this case, use **CAN 1 - user defined baud rate** to provide

# Softing CAN-AC2-ISA with Philips PCA 82C200 Setup

---

the 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 – baud rate**

Defines the most common baud rates for CAN port 2. If special timing is necessary (baud rate), the value `User` defined can be selected. In this case, use **CAN 1 - User defined baud rate** to provide the 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.

## **Initialization command structure and Termination**

Defines CAN messages sent during initialization and termination of the Setup block.

## **Memory base address**

Defines the memory base address of the board. Hardware jumpers on the board itself set the address range that the board uses. Refer to the Softing user manual on how to set the various address ranges. The setting in the dialog box must correspond to the jumper setting; otherwise the board cannot be accessed. The available address ranges (memory base address) in the pop-up menu are those supported by the board. Because the xPC Target kernel only reserves a subrange (C0000 to DC000) of the 640 KB to 1 MB address range for memory-mapped devices, the valid settings when used within a xPC target system are

## Softing CAN-AC2-ISA with Philips PCA 82C200 Setup

---

- 1 (16k): D0000-D3FFF
- 2 (16k): D4000-D7FFF

The board allows you to terminate each of the two CAN ports separately by means of hardware jumpers. Refer to the Softing user manual on how to set the jumpers. Both CAN ports must be terminated properly when you use the loop-back model provided to test the board and drivers.

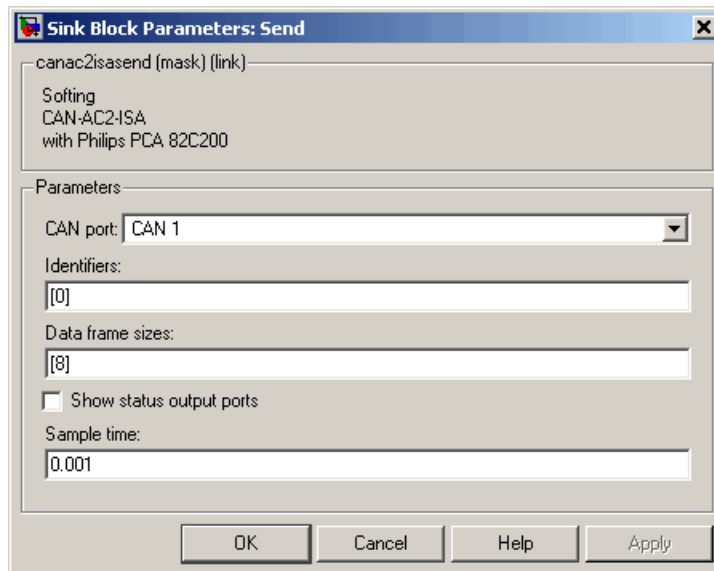


# Softing CAN-AC2-ISA with Philips PCA 82C200 Send

**Purpose** Softing CAN-AC2-ISA with Philips PCA 82C200 Send block

**Library** xPC Target Library for CAN

**Description** The Send driver block transmits data to a CAN network from within a block model.



## Block Parameters

**CAN port** Selects the CAN port to which the CAN message is sent.

**Identifiers** Defines the identifiers of the CAN messages sent by this block. It must be a row vector where the elements define a set of standard identifiers. Each element must be in the range between 0 and 2031. The number of identifiers for each CAN port in a model per physical CAN board cannot exceed 200 (limitation of the firmware's dynamic object mode). The number of elements defined here also specifies the number of input ports of the block. The

# Softing CAN-AC2-ISA with Philips PCA 82C200 Send

---

block icon displays the selected identifier at each input port. Each input port accepts the data frame to be sent along with the CAN message. The signal entering each input port must be a scalar of type double representing the maximum size of 8 bytes of a CAN message data frame.

## **Data frame sizes**

Defines the data frame size for each identifier (CAN message) in bytes. It must be a row vector in which 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 the identifiers defined in the **Identifiers** parameter must be the same, you can provide the size as a scalar only and scalar expansion applies. If the sizes are different for at least two identifiers (CAN messages), you must provide one size element for each identifier defined in the **Identifiers** parameter. Therefore the length of the two vectors must be the same.

## **Show status output ports**

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

## **Sample time**

Defines the sample time at which the Send block is executed during a model (target application) run.

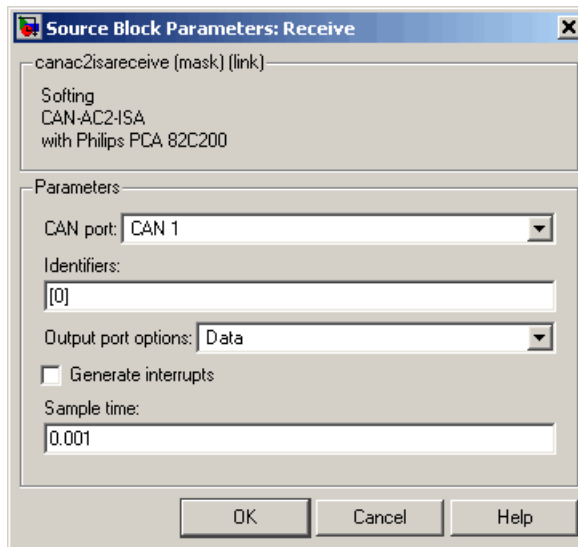
You can use as many instances of the Send block in the model as needed. For example, by using two instances of the block with different sample times, you can send CAN messages out at different rates. Or you can use multiple instances to structure your model more efficiently.

# Softing CAN-AC2-ISA with Philips PCA 82C200 Receive

**Purpose** Softing CAN-AC2-ISA with Philips PCA 82C200 Receive block

**Library** xPC Target library for CAN

**Description** The Receive driver block retrieves data from a CAN network to be used within a block model.



## Block Parameters

### CAN port

Defines the CAN port from which the CAN messages are retrieved.

### Identifiers

Defines the identifiers of the CAN messages retrieved by this block. It must be a row vector in which the elements define a set of standard identifiers. Each element must be in the range between 0 and 2031. The number of identifiers for each CAN port in a model per physical CAN board cannot exceed 200 (limitation of the firmware's dynamic object mode). The number of elements defined here, also defines the number of output ports of the block. The block icon displays the selected identifier at each output port.

# Softing CAN-AC2-ISA with Philips PCA 82C200 Receive

---

Each output port outputs the data frame being retrieved along with the CAN message. The signal leaving each output port is a scalar of type double representing the maximum size of 8 bytes of a CAN message data frame.

## **Output port options**

Defines the type of retrieved data 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 latest time at which a CAN message with the corresponding identifier was received. This time information in seconds (with a resolution of 1 microsecond) can be used to implement time-out logic within your model.

The pop-up menu lets you select the output information output at each output port of the block. If you select Data, each output port signal is a scalar only. If you select Data - Status, each output port signal is a vector with two elements in which the first element contains the data frame and the second element the status information. If you select Data - Status - Timestamp, each output port signal is a vector with three elements in which the first element contains the data frame, the second element the status information, and the third element the timestamp.

## **Generate interrupts**

Defines whether the CAN messages defined in this instance of the block initiate an interrupt from the CAN board each time they are received. If selected, you can use CAN messages to control model (target application) execution.

## **Sample time**

Defines the sample time at which the Receive block is executed during a model (target application) run.

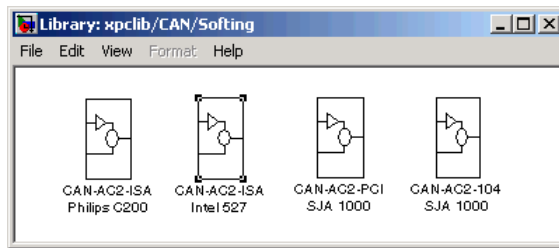
## Softing CAN-AC2-ISA with Philips PCA 82C200 Receive

---

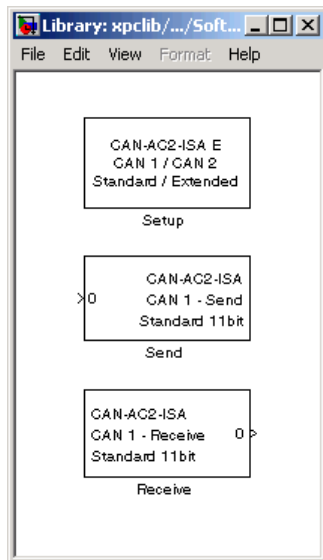
You can use as many instances of the Receive block in the model as needed. For example, by using two instances of the block with different sample times, you can retrieve CAN messages at different rates. Or you can use multiple instances to structure your model more efficiently.

## CAN Blocks for the CAN-AC2 (ISA) with Intel 82527 CAN Controller

The driver blocks described here support the CAN-AC2 (ISA) with piggyback modules. The Intel 82527 chip is used as the CAN controller in this configuration and supports both standard and extended identifier ranges in parallel. The driver block set for this board is found in the xPC Target I/O block library in the group CAN/Softing.



The second block, CAN-AC2-ISA Intel527, contains the three available CAN blocks: Setup, Send, and Receive.

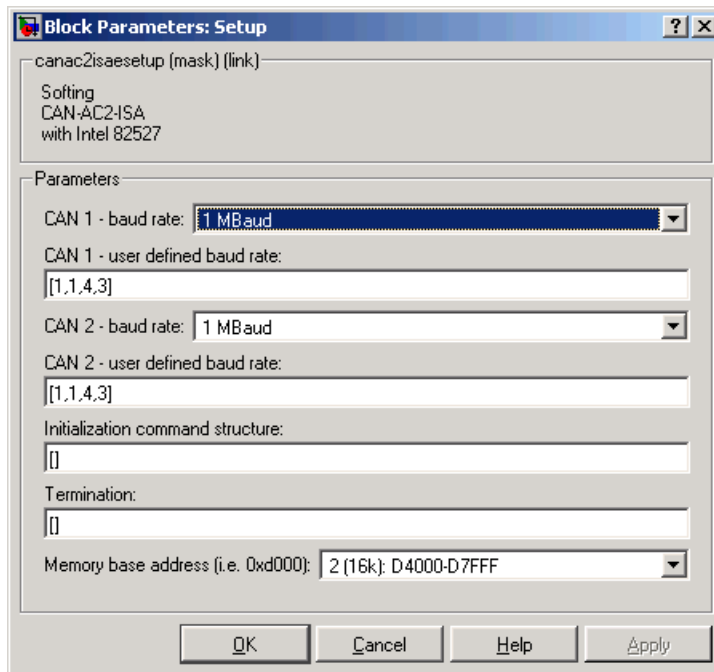


# Softing CAN-AC2-ISA with Intel 82527 Setup

**Purpose** Softing CAN-AC2-ISA with Intel 82527 Setup block

**Library** xPC Target library for CAN

**Description** The Setup block defines general settings of the installed CAN board. Because the CAN driver blocks for this board only supports a single physical board for each target system, this block can only be used once (one instance) in a model.



## Block Parameters

### CAN 1 - baud rate

Defines the most common baud rates for CAN port 1. If special timing is necessary (baud rate), the value **User defined** can be selected. In this case, use the **CAN 1 - user-defined baud rate** parameter to provide the four values for the timing information. The vector elements have the following meanings:

# Softing CAN-AC2-ISA with Intel 82527 Setup

---

[ 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 - baud rate**

Defines the most common baud rates for CAN port 1. If special timing is necessary (baud rate), the value `User` defined can be selected. In this case, use **CAN 1 - user-defined baud rate** parameter the 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.

## **Initialization command structure and Termination**

Define CAN messages sent during initialization and termination of the Setup block.

## **Memory base address**

Defines the memory base address of the board. The address range used by the board must be set by hardware jumpers on the board itself. Refer to the Softing user manual on how to set the various address ranges. The setting in the dialog box must correspond to the jumper setting; otherwise the board cannot be accessed. The available address ranges (memory base address) in the pop-up menu are those supported by the board. Because the xPC Target kernel only reserves a subrange (C0000 to DC000) of the 640 KB to 1 MB address range for memory-mapped devices, the valid settings when used within a xPC Target system only are

- 1 (16k): D0000-D3FFF
- 2 (16k): D4000-D7FFF



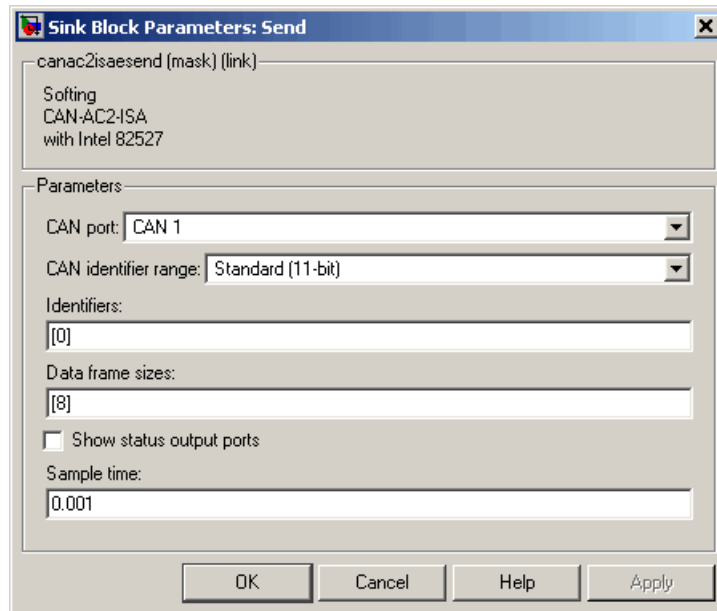
## Softing CAN-AC2-ISA with Intel 82527 Setup

---

The board allows you to terminate each of the two CAN ports separately by means of hardware jumpers. Refer to the Softing user manual on how to set the jumpers. Both CAN ports must be terminated properly when you use the loop-back model provided to test the board and drivers.

# Softing CAN-AC2-ISA with Intel 82527 Send

- Purpose** Softing CAN-AC2-ISA with Intel 82527 Send block
- Library** xPC Target Library for CAN
- Description** The Send driver block transmits data to a CAN network from within a block model.



## Block Parameters

- CAN port** Selects the CAN port to which the CAN message is sent.
- CAN identifier range** Selects the identifier range of the CAN messages sent by this block instance. If an application makes use of mixed standard and extended identifier ranges, you must use at least two instances of this block, each defining the corresponding identifier range.

## **Identifiers**

Defines the identifiers of the CAN messages sent by this block. It must be a row vector in which 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 (limitation of the firmware's dynamic object mode). The number of elements defined here also defines the number of input ports of the block. The block icon displays the selected identifier at each input port. Each input port accepts the data frame to be sent along with the CAN message. The signal entering each input port must be a scalar of type double representing the maximum size of 8 bytes of a CAN message data frame.

## **Data frame sizes**

Defines the data frame size for each identifier (CAN message) in bytes. It must be a row vector where the elements define a set of data frame sizes. Each element must be in the range between 1 and 8. If the data frame sizes for all identifiers defined in the **Identifiers** parameter must be the same, you can provide the size as a scalar only and scalar expansion applies. If the sizes are different for at least two identifiers (CAN messages), you can provide one size element for each identifier defined in the **Identifiers** parameter. Therefore the length of the two vectors must be the same.

## **Show status: Output ports**

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

## **Sample time**

Defines the sample time at which the Send block is executed during a model (target application) run.

## Softing CAN-AC2-ISA with Intel 82527 Send

---

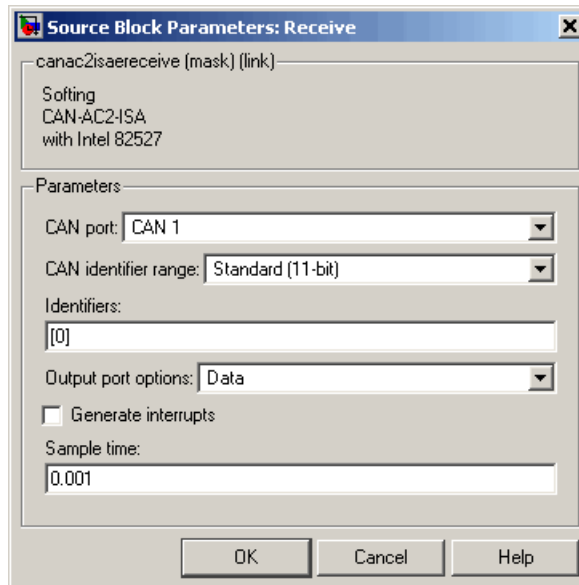
You can use as many instances of the Send block in the model as needed. For example, by using two instances of the block, you can send CAN messages at different sample times. Or you can use multiple instances to structure your model more efficiently.

# Softing CAN-AC2-ISA with Intel 82527 Receive

**Purpose** Softing CAN-AC2-ISA with Intel 82527 Receive block

**Library** xPC Target Library for CAN

**Description** The Receive driver block retrieves data from a CAN network to be used within a block model.



## Block Parameters

**CAN port** Selects the CAN port from which to retrieve the CAN messages.

**CAN identifier range** Selects the identifier range of the CAN messages retrieved by this block instance. If an application makes use of mixed standard and extended identifier ranges, you must use at least two instances of this block, each defining the corresponding identifier range.

## Identifiers

Defines the identifiers of the CAN messages retrieved by this block. It must be a row vector in which 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 (limitation of the firmware's dynamic object mode). The number of elements defined here also defines the number of output ports of the block. The block icon displays the selected identifier at each output port. Each output port outputs the data frame being retrieved along with the CAN message. The signal leaving each output port is a scalar of type double representing the maximum size of 8 bytes of a CAN message data frame.

## Output port options

Defines the type of retrieved data 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 latest time at which a CAN message with the corresponding identifier was received. This time information in seconds (with a resolution of 1 microsecond) can be used to implement time-out logic within your model.

The pop-up menu lets you select the output information output at each output port of the block. If you select Data, each output port signal is a scalar only. If you select Data - Status, each output port signal is a vector with two elements in which the first element contains the data frame and the second element the status information. If you select Data - Status - Timestamp, each output port signal is a vector with three elements in which the first element contains the data frame, the second element the status information, and the third element the timestamp.

## **Generate interrupts**

Defines whether the CAN messages defined in this instance of the block initiate an interrupt from the CAN board each time they are received. If selected, you can use CAN messages to control model (target application) execution.

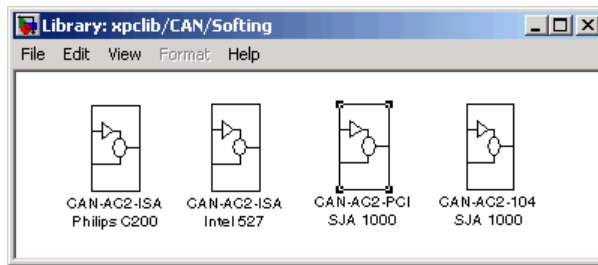
## **Sample time**

Defines the sample time at which the Send block is executed during a model (target application) run.

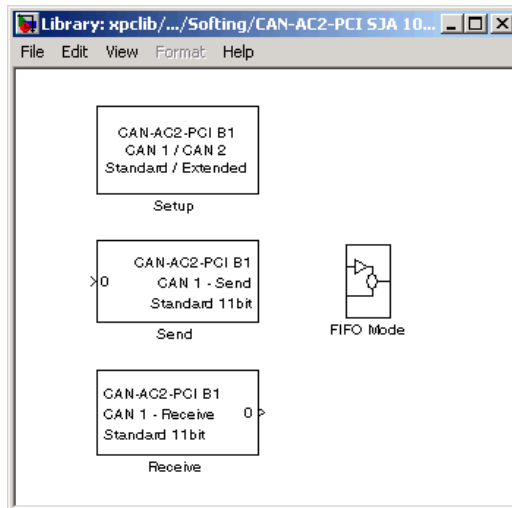
You can use as many instances of the Receive block in the model as needed. For example, by using two instances of the block, you can send CAN messages at different sample times. Or you can use multiple instances to structure your model more efficiently.

## CAN Blocks for the CAN-AC2-PCI with Philips SJA1000 CAN Controller

The driver blocks described here support the CAN-AC2-PCI. The Philips SJA1000 chip is used as the CAN controller in this configuration and supports both standard and extended identifier ranges in parallel. The driver block set for this board is found in the xPC Target I/O block library in the group CAN/Softing.



The third block group, CAN-AC2-PCI SJA 1000, contains the three available CAN blocks: Setup, Send, and Receive, plus a FIFO Mode block, which is discussed in Chapter 5, "CAN I/O Support for FIFO".



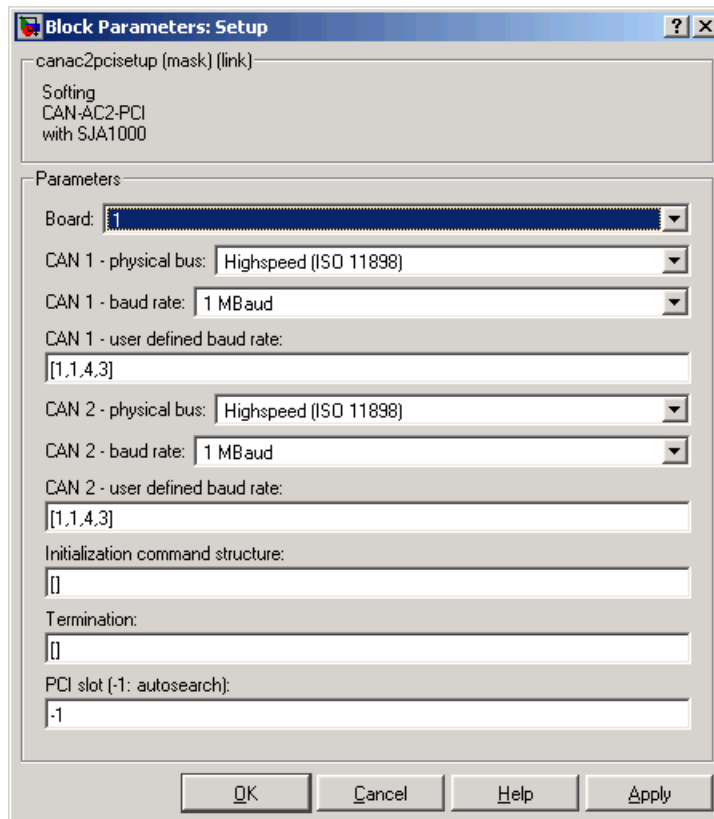


# Softing CAN-AC2-PCI with SJA1000 Setup

**Purpose** Softing CAN-AC2-PCI with SJA1000 Setup block

**Library** xPC Target Library for CAN

**Description** The Setup block defines general settings of the installed CAN boards. The CAN driver blocks for this board support up to three boards for each target system, making up to six CAN ports available. For each board in the target system, you must use exactly one Setup driver block.



# Softing CAN-AC2-PCI with SJA1000 Setup

---

## Block Parameters

### Board

Defines the board being accessed by this driver block instance. If multiple boards are present in the target PC, you can use the board number (1...3) to differentiate the boards. The physical board referenced by the board number depends on the **PCI Slot** parameter. If just one board is present in the target system, select board number 1.

### CAN 1 - physical bus

Defines the physical CAN bus type of CAN port 1. In the board's standard hardware configuration, only high-speed CAN is supported. By extending the board with low-speed CAN piggyback modules, you can also select low-speed CAN as the physical bus. Do not change this value to low-speed if no module is present for the corresponding CAN port. If the module is present (see the Softing user manual on how to install the modules), you can select between high-speed and low-speed CAN here.

### CAN 1- baud rate

Defines the most common baud rates for CAN port 1. If special timing is necessary (baud rate), you can select the value User defined. In this case, you use the **CAN 1 - user defined baud rate** parameter to provide the 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 - physical bus

Defines the physical CAN bus type of CAN port 2. In the board's standard hardware configuration, only high-speed CAN is supported. By extending the board with low-speed CAN piggyback modules, you can also select low-speed CAN as the physical bus. Do not set this value should to low-speed if no module is present for the corresponding CAN port. If the module is present (see the

# Softing CAN-AC2-PCI with SJA1000 Setup

---

Softing user manual on how to install the modules), you can select between high-speed and low-speed CAN here.

## **CAN 2 - baud rate**

Defines the most common baud rates for CAN port 2. If special timing is necessary (baud rate), the value User defined can be selected. In this case, you can use the **CAN 2 - user defined baud rate** parameter to provide the 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.

## **Initialization command structure and Termination**

Defines CAN messages sent during initialization and termination of the Setup block.

## **PCI slot (-1: autosearch)**

Defines the PCI slot in which the referenced board (board number) resides. If only one board is present in the target system, set the value for this control to -1 (autosearch). This ensures that the xPC Target kernel automatically finds the board regardless of the PCI slot it is plugged into. If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. Use the xPC Target function `getxpcpci` to query the target system for installed PCI boards and the PCI slots they are plugged into. For more information see `help getxpcpci`.

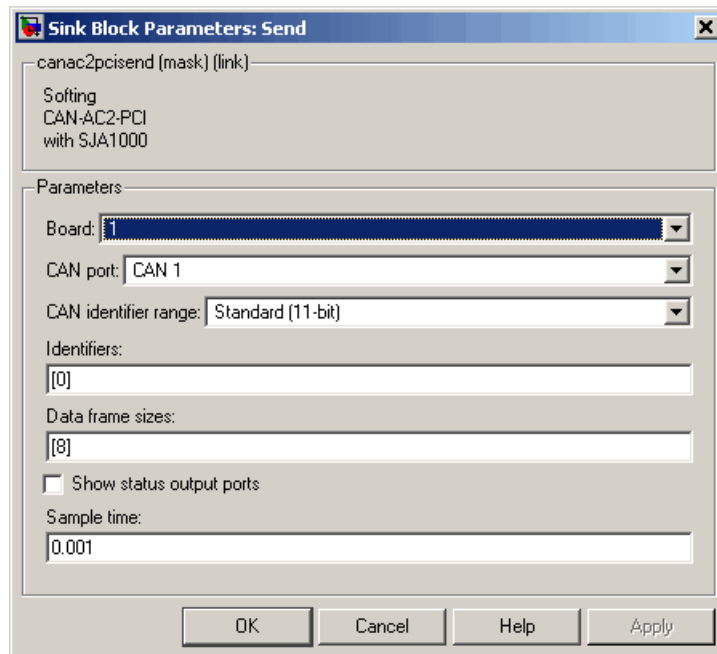
The board allows you to terminate each of the two CAN ports separately by means of DIP switches at the rear panel. Refer to the Softing user manual on how to set the DIP switches. Both CAN ports must be terminated properly when you use the loop-back model provided to test the board and drivers.

# Softing CAN-AC2-PCI with SJA1000 Send

**Purpose** Softing CAN-AC2-PCI with SJA1000 Send block

**Library** xPC Target Library for CAN

**Description** The Send driver block transmits data to a CAN network from within a block model.



## Block Parameters

### Board

Defines the board used to send out the CAN messages defined by this block instance. For more information about the meaning of the board number see the Setup driver block described above. If just one board is present in the target system, select board number 1.

## **CAN port**

Selects the CAN port to which the CAN message is sent.

## **CAN identifier range**

Selects the identifier range of the CAN messages sent by this block instance. If an application makes use of mixed standard and extended identifier ranges, you must use at least two instances of this block, each defining the corresponding identifier range.

## **Identifiers**

Defines the identifiers of the CAN messages sent by this block. It must be a row vector in which 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 (limitation of the firmware's dynamic object mode). The number of elements defined here also define the number of input ports of the block. The block icon displays the selected identifier at each input port. Each input port accepts the data frame to be sent along with the CAN message. The signal entering each input port must be a scalar of type double representing the maximum size of 8 bytes of a CAN message data frame.

## **Data frame sizes**

Defines the data frame size for each identifier (CAN message) in bytes. It must be a row vector in which 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 control above must be the same, you can provide the size as a scalar only and scalar expansion applies. If the sizes are different for at least two identifiers (CAN messages), you must provide one size element for each identifier defined in the **Identifiers** parameter. Therefore the lengths of the two vectors must be the same.

## **Show status output ports**

Enables status output ports for each identifier (CAN message). If the check box is checked the block shows as many output ports

## Softing CAN-AC2-PCI with SJA1000 Send

---

as input ports. The data type of each output port is a double and the value is identical to the return argument of function `CANPC_write_object(...)`, described in the Softing user manual. Refer to the manual for more information.

### **Sample time**

Defines the sample time at which the Send block is executed during a model (target application) run.

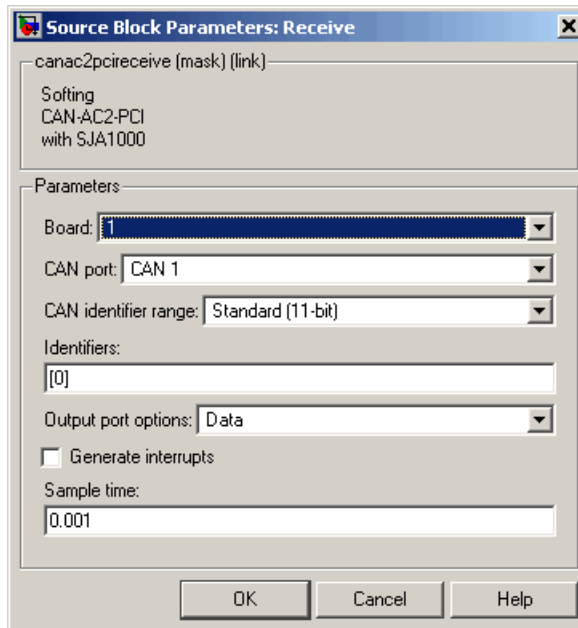
You can use as many instances of the Send block in the model as needed. For example, by using two instances of the block, you can send CAN messages at different sample times. Or you can use multiple instances to structure your model more efficiently.

# Softing CAN-AC2-PCI with SJA1000 Receive

**Purpose** Softing CAN-AC2-PCI with SJA1000 Receive block

**Library** xPC Target Library for CAN

**Description** The Receive driver block retrieves 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.



## Block Parameters

### Board

Defines the board the CAN messages defined by this block instance are retrieved from. 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.

# Softing CAN-AC2-PCI with SJA1000 Receive

---

## **CAN port**

Selects the CAN port from which the CAN messages are retrieved.

## **CAN identifier range**

Selects the identifier range of the CAN messages retrieved by this block instance. If an application makes use of mixed standard and extended identifier ranges, you must use at least two instances of this block, each defining the corresponding identifier range.

## **Identifiers**

Defines the identifiers of the CAN messages retrieved by this block. It must be a row vector in which 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 (limitation of the firmware's dynamic object mode). The number of elements defined here also defines the number of output ports of the block. The block icon displays the selected identifier at each output port. Each output port outputs the data frame being retrieved along with the CAN message. The signal leaving each output port is a scalar of type double representing the maximum size of 8 bytes of a CAN message data frame.

## **Output port options**

Defines the type of retrieved data 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 latest time at which a CAN message with the corresponding identifier was received. This time information in seconds (with a resolution of 1 microsecond) can be used to implement time-out logic within your model. The pop-up menu lets you select the output information at each output port of the block. If you select Data, each output port signal is a scalar only. If you select Data - Status, each



# Softing CAN-AC2-PCI with SJA1000 Receive

---

output port signal is a vector with two elements, in which the first element contains the data frame and the second element the status information. If you select Data - Status - Timestamp, each output port signal is a vector with three elements, in which the first element contains the data frame, the second element the status information, and the third element the timestamp.

## **Generate interrupts**

Defines whether the CAN messages defined in this instance of the block will initiate an interrupt from the CAN board each time they are received. If selected, you can use CAN messages to control model (target application) execution.

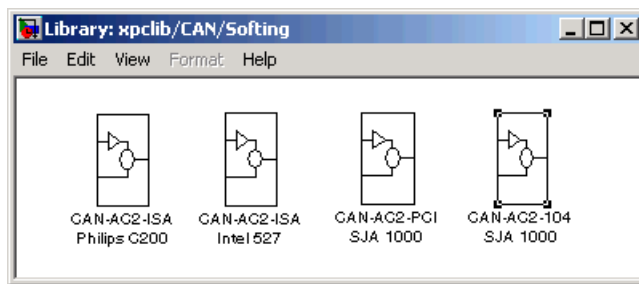
## **Sample time**

Defines the sample time at which the Send block is executed during a model (target application) run.

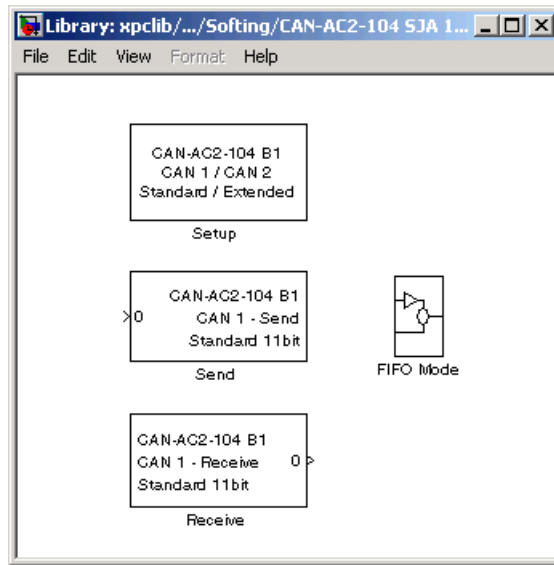
You can use as many instances of the Send block in the model as needed. For example, by using two instances of the block, you can send CAN messages at different sample times. Or you can use multiple instances to structure your model more efficiently.

# CAN Blocks for the CAN-AC2-104 (PC/104) with Philips SJA1000 CAN Controller

The driver blocks described here support the CAN-AC2-104 (PC/104). The Philips SJA1000 chip is used as the CAN controller in this configuration and supports both standard and extended identifier ranges in parallel. The driver block set for this board is found in the xPC Target I/O block library in the group CAN/Softing.



The fourth block group, CAN-AC2-104 SJA 1000, contains the three available CAN blocks: Setup, Send, and Receive, plus a FIFO Mode block, which is discussed in Chapter 5, "CAN I/O Support for FIFO".

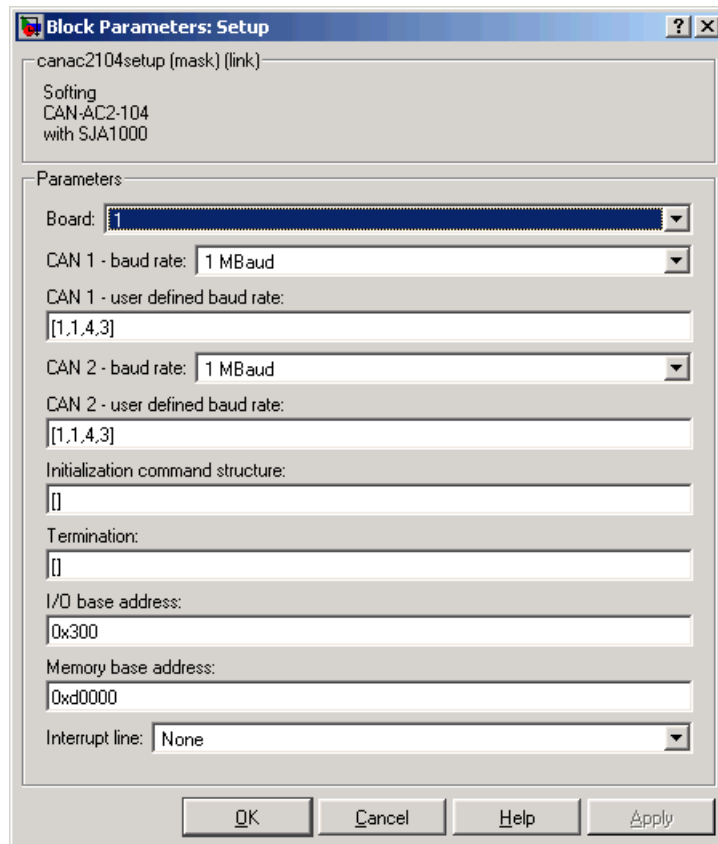


# Softing CAN-AC2-104 with SJA1000 Setup

**Purpose** Softing CAN-AC2-104 with SJA1000 Setup block

**Library** xPC Target Library for CAN

**Description** The Setup block defines general settings of the stacked CAN boards. The CAN driver blocks for this board support up to three boards for each target system, making up to six CAN ports available. For each board in the target system, you must use exactly one Setup driver block.



## Block Parameters

### Board

Defines the board being accessed by this driver block instance. If multiple boards are present in the target PC, you can use the board number (1...3) to differentiate the boards. The physical board referenced by the board number depends on the **PCI Slot** parameter. If just one board is present in the target system, select board number 1. The physical board referenced by the board number depends on the **I/O base address** parameter.

### CAN 1 - baud rate

Defines the most common baud rates for CAN port 1. If special timing is necessary (baud rate), you can select the value User defined. In this case, use the **CAN 1 - user defined baud rate** parameter to provide the 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 - baud rate

Defines the most common baud rates for CAN port 2. If special timing is necessary (baud rate), you can select the value User defined. In this case, use the **CAN 2 - user defined baud rate** parameter to provide the 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.

### Initialization command structure and Termination

Define CAN messages sent during initialization and termination of the Setup block.

# Softing CAN-AC2-104 with SJA1000 Setup

---

## **I/O base address**

Defines the I/O base address of the board to be accessed by this block instance. The I/O base address is given by the DIP switch setting on the board itself. The I/O address range is 3 bytes and is mainly used to transfer the memory base address the board should use. See the Softing user manual for this board to set the I/O base address. The I/O base address entered in this control must correspond with the DIP switch setting on the board. If more than one board is present in the target system, a different I/O base address must be entered for each board. In this case the I/O base address itself defines which board is referenced by which board number.

## **Memory base address**

Defines the memory base address of the board to be accessed by this block instance. The memory base address is a software setting only (no corresponding DIP switch is found on the board). The memory address range is 4 KB. If more than one board is present in the target system, a different memory base address must be entered for each board. You must make sure that the defined address ranges do not overlap. Because the xPC Target kernel only reserves a subset of the address range between 640 KB and 1 MB for memory mapped devices, the address ranges must be within the following range:

C0000 - DC000

The board allows you to terminate 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 must be set. Both CAN ports must be terminated properly when you use the loop-back model provided to test the board and drivers.

## **Interrupt line**

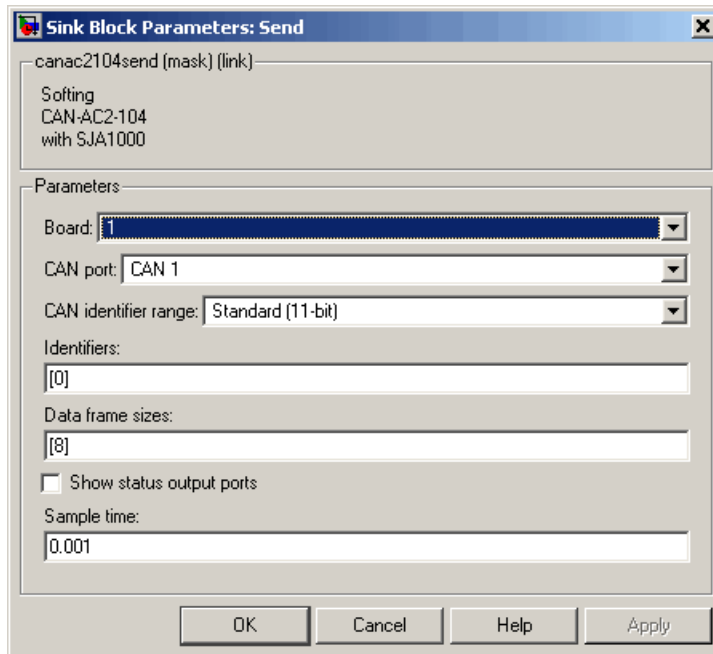
Selects an interrupt line from the list.

# Softing CAN-AC2-104 with SJA1000 Send

**Purpose** Softing CAN-AC2-104 with SJA1000 Send block

**Library** xPC Target Library for CAN

**Description** The Send driver block transmits data to a CAN network from within a block model. You can define up to 200 send objects for standard and extended identifiers for each CAN channel.



## Block Parameters

### Board

Defines the 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.

# Softing CAN-AC2-104 with SJA1000 Send

---

## **CAN Port**

Selects the CAN port to send the CAN message.

## **CAN identifier range**

Selects the identifier range of the CAN messages sent by this block instance. If an application makes use of mixed standard and extended identifier ranges, you must use at least two instances of this block, each defining the corresponding identifier range.

## **Identifiers**

Defines the identifiers of the CAN messages sent by this block. It must be a row vector in which 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 also defines the number of input ports of the block. The block icon displays the selected identifier at each input port. Each input port accepts the data frame to be sent along with the CAN message. The signal entering each input port must be a scalar of type double representing the maximum size of 8 bytes of a CAN message data frame.

## **Data frame sizes**

Defines the data frame size for each identifier (CAN message) in bytes. It must be a row vector in which 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 must be the same, you can provide the size 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 **Identifiers** control. Therefore the lengths of the two vectors must be the same.

## **Show status output ports**

Enables 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



# Softing CAN-AC2-104 with SJA1000 Send

---

`CANPC_write_object(...)`, described in the Softing user manual.  
Refer to the manual for more information.

## **Sample time**

Defines the sample time at which the Send block is executed during a model (target application) run.

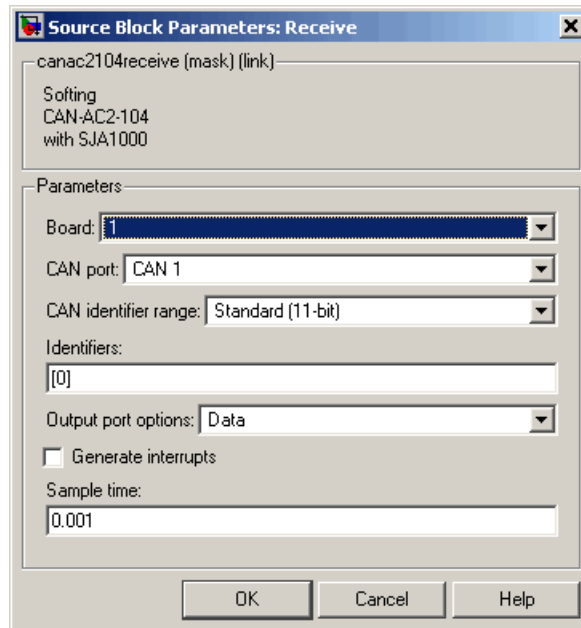
You can use as many instances of the Send block in the model as needed. For example, by using two instances of the block, you can define different sample times at which CAN messages are sent out. Or you can use multiple instances to structure your model more efficiently.

# Softing CAN-AC2-104 with SJA1000 Receive

**Purpose** Softing CAN-AC2-104 with SJA1000 Receive block

**Library** xPC Target Library for CAN

**Description** The Receive driver block retrieves 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.



## Block Parameters

### Board

Defines the board from which the CAN messages defined by this block instance are to be retrieved. For more information about the meaning of the board number, see the Setup driver block. If just one board is present in the target system, select board number 1.

### CAN Port

Selects the CAN port from which to retrieve the CAN message.

## **CAN identifier range**

Selects the identifier range of the CAN messages retrieved by this block instance. If an application makes use of mixed standard and extended identifier ranges, at least two instances of this block must be used, each defining the corresponding identifier range.

## **Identifiers**

Specifies the identifiers of the CAN messages retrieved by this block. It must be a row vector where the elements define a set of either standard or extended identifiers. Each element must be in the range between 0 and 2031 for standard identifiers, or 0 and  $2^{29} - 1$  for extended identifiers. The number of identifiers for each CAN port in a model per physical CAN board cannot exceed 200. The number of elements defined here defines the number of output ports of the block. The block icon displays the selected identifier at each output port. Each output port outputs the data frame being retrieved along with the CAN message. The signal leaving each output port is a scalar of type double representing the maximum size of 8 bytes of a CAN message data frame.

## **Output port options**

Defines the type of retrieved data 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 was received. This time information in seconds (with a resolution of 1 microsecond) can be used to implement time-out logic within your model.

The pop-up menu lets you select the output information output at each output port of the block. If you select `Data`, each output port signal is a scalar only. If you select `Data-Status`, each output port signal is a vector with two elements, in which the first element contains the data frame and the second element the

# Softing CAN-AC2-104 with SJA1000 Receive

---

status information. If you select Data-Status-Timestamp, each output port signal is a vector with three elements, in which the first element contains the data frame, the second element the status information, and the third element the timestamp.

## **Generate interrupts**

Defines whether the CAN messages defined in this instance of the block initiate an interrupt from the CAN board each time they are received. If selected, you can use CAN messages to control model (target application) execution.

## **Sample time**

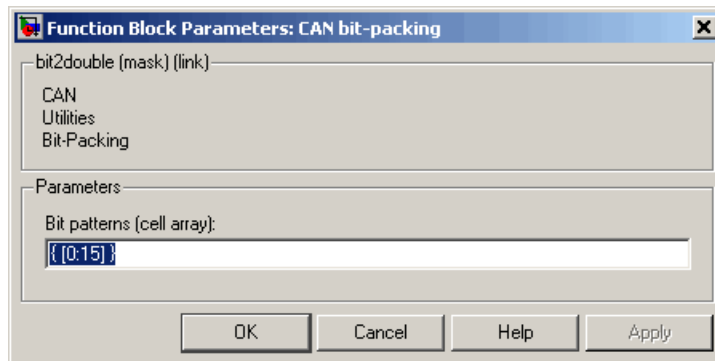
Defines the sample time at which the Send block is executed during a model (target application) run.

You can use as many instances of the Receive 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 retrieved. Or you can use multiple instances to structure your model more efficiently. You can define up to 200 receive objects for standard and extended identifiers for each CAN channel.

**Purpose** CAN Bit-Packing block

**Library** xPC Target Library for CAN

**Description** This block constructs CAN data frames, and its output port is normally connected to an input port of a CAN Send driver block. The block has one output port of data type double (a scalar), which represents the data frame entity constructed by the signals entering the block at its input ports. The number of input ports depends on the setting in the block's dialog box.



## Block Parameters

### Bit Patterns

Specify bit patterns. The data type entered in the control must be a MATLAB cell array vector. The number of elements in the cell array define the number of input ports shown by this block instance. The cell array elements must be of type double array and define the position of each bit of the incoming value (data typed input port) in the outgoing double value (data frame).

From a data type perspective (input ports), the block behaves like a Simulink Sink block, and therefore the data types of the input ports are inherited from the driving blocks.

# CAN Bit-Packing

---

The sample time of the block is also inherited from the driving blocks. Therefore no explicit sample time must be provided in the block's dialog box.

## Example

The functionality of the block is best explained by means of an example.

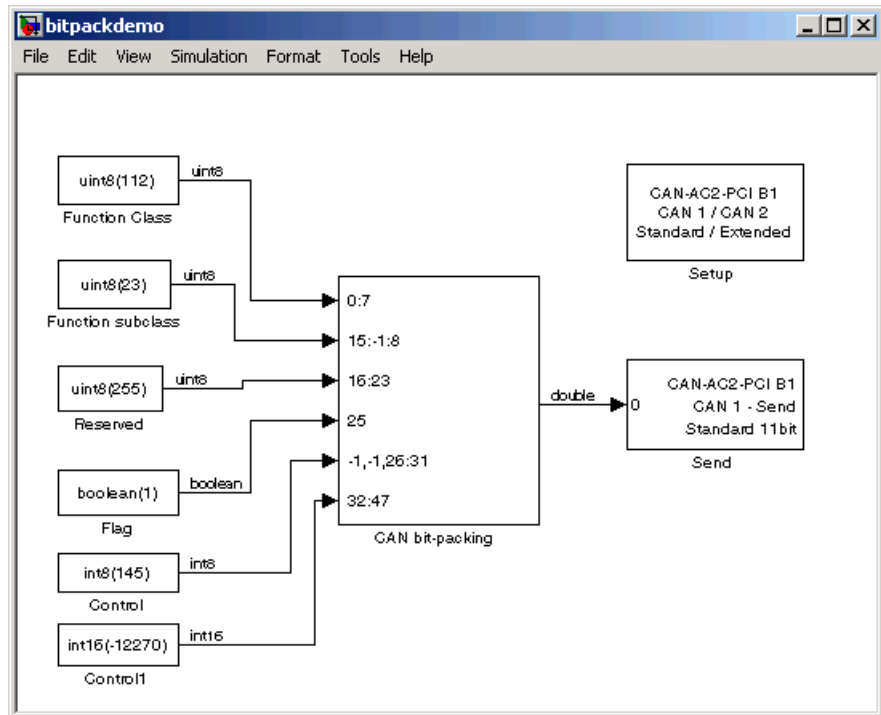
Assume that a node on the CAN network needs to receive a CAN message with identifier 156 having the following data frame content. The data frame must be 6 bytes long.

Byte 0	Function class of type uint8
Byte 1	Function subclass of type uint8 with reversed bit order
Byte 2	Reserved, all bits must be 1
Byte 3	Bit 0 must be 0, Bit 1 must be a boolean (flag), bits 2 to 7 must be bit 2 to 7 of an incoming int8 value (control)
Byte 4 and 5	Value of type int16

The bit pattern cell array, which bit-packs the data frame according to the above specification, can look as follows:

```
{ [0:7] , [15:-1:8] , [16:23] , [25] , [-1,-1,26:31] , [32:47] }
```

And the Simulink model simulating the needed behavior would be as shown.



Analyze the model.

The first input is the function class of type uint8, which has an example value of 112. This value becomes byte 0 (bits 0 to 7) of the data frame. Therefore the first bit (element 1 of double array [0:7]) gets bit 0 of the data frame, the second bit 1, and so on. It is easiest to define this mapping by the MATLAB colon operator (:).

The second input is the function subclass of type uint8, which has an example value of 23. This value becomes byte 1 (bits 8:15) of the data frame but in reversed bit order. Therefore the first bit (element 1 of double array [15:-1:8]) gets bit 15, the second bit 14, and so on. It is

# CAN Bit-Packing

---

easiest to define this mapping by the MATLAB colon operator (:) and an increment of -1.

The third input is only necessary because the reserved byte 2 must have all bits set to 1. If a bit position in the outgoing data frame is not referenced by a bit pattern array element, the bit is 0 by default, but there is no way to set them to 1 as the default. Therefore a uint8 constant with value 255 must be brought in externally. The constant 255 must get to bit position 16 to 23 (byte 2) of the outgoing data frame.

Because bit 0 of data frame byte 3 (bit 24) must be 0, and 0 is the default bit value if not referenced by a bit pattern array element, no explicit action is taken here.

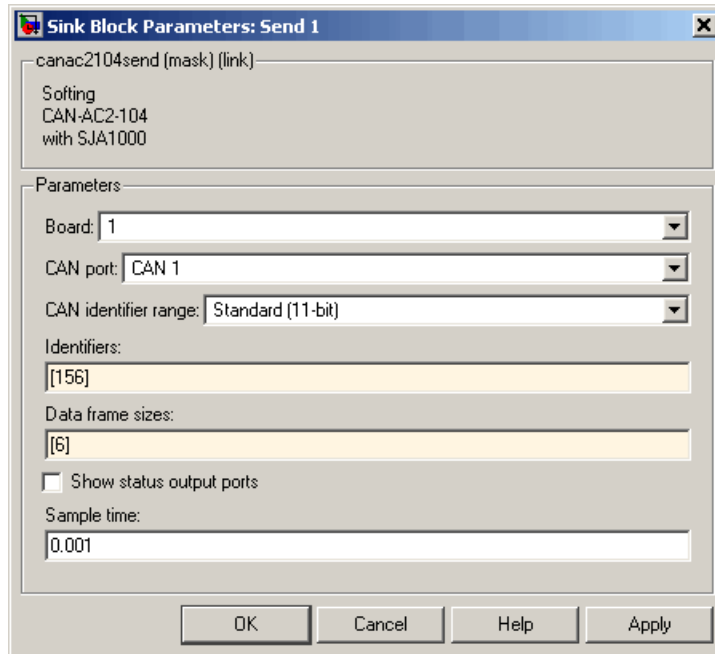
The fourth input is the flag of type Boolean, which has an example value of 1. This value must become bit 1 of byte 3 (bit 25) of the data frame. Therefore the single bit (element 1 of double array [25]) must get bit 25 of the data frame.

The fifth input is the control of type int8, which has an example value of 121. But only bits 2 to 7 must be mapped into the outgoing data frame or, in other words, bits 0 and 1 must be thrown away. Because indexing of incoming values always starts with the first bit (bit 0), a special indexing value (-1) must be used to skip bit 0 and 1 of the incoming int8 value. Bits 2 to 7 are directly mapped to bits 2 to 7 of byte 3 (bits 26 to 31) of the outgoing data frame. This leads to the following bit pattern: [-1,-1,26:31].

The sixth input is the value of type int16, which has an example value of -12270. This value must become byte 4 and 5 (bits 32 to 47) of the outgoing data frame. Therefore the first bit (element 1 of double array [32:47]) must get bit 32 of the data frame, the second bit 33, and so on. It is easiest to define this mapping by the MATLAB colon operator (:).



The output of the block then consists of a double value representing the packed data types within the first six bytes. The last two bytes are zero. This means that even in the case where less than eight bytes are significant, the CAN data frame is always represented by a double value (eight bytes). The value of the constructed floating-point double does not have any particular meaning but you still see it with a numerical display.



The data frame is then propagated to the CAN Send driver block and is sent as part of a CAN message having identifier 156. In the Send block's dialog box, the data frame size is defined as 6 bytes. This ensures that only the first six bytes of the incoming double value are transmitted as part of the CAN message.

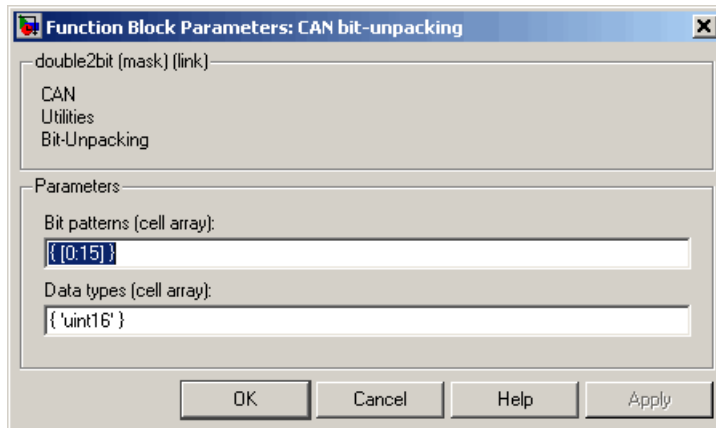
# CAN Bit-Unpacking

---

**Purpose** CAN Bit-Unpacking block

**Library** xPC Target Library for CAN

**Description** This block is used to extract CAN data frames, and its input port is normally connected to an output port of a CAN Receive driver block. The block has one input port of data type double (a scalar), which represents the data frame entity from which the signals are extracted and leaving the block at its output ports. The number of output ports and the data type of each output port depend on the settings in the block's dialog box.



## Block Parameters

### Bit Patterns

Lets you define the bit patterns in a flexible way. The data type entered in the control must be a MATLAB cell array vector. The number of elements in the cell array define the number of output ports shown by this block instance. The cell array elements must be of type double array and define the position of each bit of the incoming value (data typed output port) in the incoming double value (data frame).

## Data Types

From a data type perspective (output ports), the block behaves like a Simulink Source block, and therefore the data types of the output ports must be defined in the second control (edit field). The data type entered in that control must be a MATLAB cell array vector of the same length as the bit pattern cell array. The cell array elements must be of type char and define the data type of the corresponding output port. The following values are supported:

```
boolean, int8, uint8, int16, uint16, int32, uint32
```

The sample time of the block is inherited from the driving block. Therefore no explicit sample time need be provided in the block's dialog box.

Note that if you unpack the data frame into a signed type (int8, int 16, or int 32), the block performs sign extension as necessary. For example, if the bit pattern is [0:4], and the data type is int8, you are extracting 5 bits into an 8 bit wide signed type. In this case, bits 5, 6, and 7 are the same as bit 4, resulting in the proper sign extension. This functionality enables you to pack and unpack negative numbers without losing accuracy. In the preceding example, you can pack and unpack numbers in the range [-16 : 15] (a fictitious int5 type).

## Example

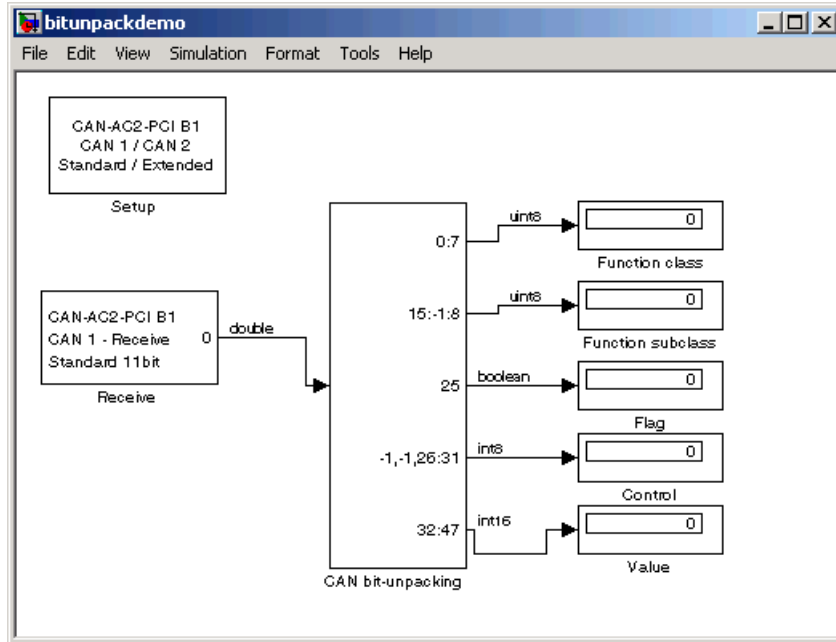
The functionality of the block is easiest explained by means of an example. The same example as used above demonstrates the functionality of the bit-packing block. But in this case, the data frame is sent by an external CAN node and is received by the target application running on an xPC Target system. Therefore the bit-unpacking block is used to extract the various data fields from the entire data frame. Because the bit pattern definitions of the packing and unpacking block are symmetric, the bit pattern definition could look exactly the same. There is one simple optimization possible: You do not need to extract byte 2 (reserved area), because its content is known. The bit pattern edit field can therefore look as follows:

```
{ [0:7] , [15:-1:8] , [25] , [-1,-1,26:31] , [32:47] }
```

# CAN Bit-Unpacking

and the data type edit field as

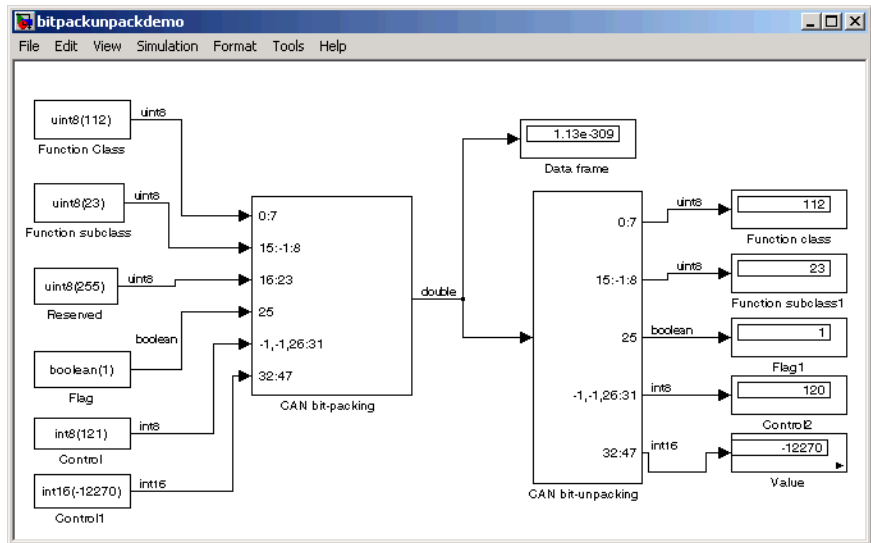
```
{ 'uint8' , 'uint8' , 'boolean' , 'int8' , 'int16' }
```



This leads to the following Simulink model.

In many cases it makes sense to test the proper bit-packing and bit-unpacking operations in a Simulink model (simulation) before building the target application. Both blocks work the same way either in Simulink or the generated code. By combining the two models shown so far, a third model emerges that can be used to simulate the behavior.

# CAN Bit-Unpacking



# CAN Timeout Detection

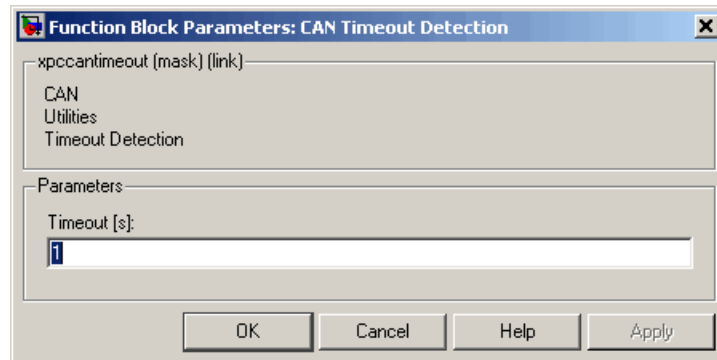
---

**Purpose** CAN Timeout Detection block

**Library** xPC Target Library for CAN

**Description** The CAN Timeout Detection block uses the timestamp information to calculate the time-out condition. For examples of this block usage, see

- “Detecting Time-Outs When Receiving CAN Messages” on page 4-16 — Detecting time-outs example
- `xpccanpci` — Loop-back example for the CAN-AC2-PCI board
- `xpccanpc104` — Loop-back example for the CAN-AC2-104 board



## Block Parameters

### Timeout

Specify the time-out value, in seconds. The output of the block is:

- 0, if no time-out has been detected
- 1, if a time-out has been detected

# CAN I/O Support for FIFO

---

This chapter includes the following sections:

Introduction (p. 5-2)

This chapter describes the alternative First In First Out (FIFO) CAN drivers provided with xPC Target.

Acceptance Filters (p. 5-6)

The CAN controller's acceptance filters can be used to ensure that certain received messages referenced by their identifiers get written into the receive FIFO.

Examples (p. 5-8)

Examples involving FIFO CAN drivers.

CAN FIFO Blocks for the CAN-AC2-PCI with Philips SJA1000 CAN Controller (p. 5-17)

The driver blocks described here support the CAN-AC2-PCI using FIFO mode.

CAN FIFO Blocks for the CAN-AC2-104 with Philips SJA1000 CAN Controller (p. 5-39)

The driver blocks described here support the CAN-AC2-104 (PC/104) using FIFO mode.

## Introduction

This chapter describes the alternative First In First Out (FIFO) CAN drivers provided with xPC Target. The standard CAN drivers for the CAN boards from Softing GmbH (<http://www.softing.com>) program the CAN board firmware to run in Dynamic Object Buffer (DOB) mode. This mode is best suited for real-time environments where it is mandatory that the driver latency time is time deterministic. Actually, running the firmware in Dynamic Object Buffer mode is always the best choice except for the undesired side effect of high driver latency times.

- **Sending a CAN message** — When sending a CAN message, the latency time is the time interval between the time accessing the board to provide all the information for the CAN message to be sent and the time the board returns the acknowledgment that the information has been received by the firmware.
- **Receiving a CAN message** — When receiving a CAN message, the latency time is the time interval between the time accessing the board to ask for current data (object data) of a certain CAN identifier and the time the board returns the actual data and other information about the CAN message.

**Disadvantages of Dynamic Object Buffer mode** — These latency times are mainly defined by the reaction time of the board firmware. In the case of the Softing boards, the latency time is the same for sending and receiving messages, with a fixed value of about 40  $\mu$ s. If your xPC Target application has to send and receive a large number of CAN messages, the overall latency time can quickly become high and can make it impossible to run the application at the desired base sample time.

For example, assuming that a specific xPC Target application gets data from 12 CAN identifiers and transmits data by using eight CAN messages, the total number of CAN board read and write accesses adds up to 20. This results in a total CAN I/O latency time of

$$20 * 40 \text{ s} = 800 \text{ s}$$

With such an application, base sample times below 800  $\mu$ s are impossible even if the dynamics of the corresponding Simulink model are simple and would only need 20  $\mu$ s of computation time.



**Advantages of Dynamic Object Buffer mode** — However, even if the CAN I/O latency time in Dynamic Object Buffer mode is high, the benefit of this mode is that the latency time stays constant almost independent of the traffic volume on the CAN network. This means that the Dynamic Object Buffer mode is best suited for xPC Target applications that only deal with a small subset of all CAN messages going over the CAN network.

## **FIFO Mode Drivers for CAN Boards from Softing**

The CAN boards from Softing support another mode, called First In First Out (FIFO) mode. In this mode the Dynamic Object Buffer mode abstraction layer in the firmware is missing, and the firmware plays the role of a slim interface between the receive and transmit FIFOs and the drivers in the application code. Because of this slimmer interface, the I/O latency times are considerably smaller. Writing to the transmit FIFO takes 4  $\mu$ s per CAN message and reading one event (CAN message) from the receive FIFO takes 17  $\mu$ s. Both of these latency times are smaller than the 40  $\mu$ s for Dynamic Object Buffer mode. While writing to the transmit FIFO is efficient, this is not the case for reading from the receive FIFO. Because the receive FIFO is filled with all CAN messages (identifiers) going over the CAN network, there can be a lot of data (CAN messages) that have to be read out of the FIFO even if their data is not used in the target application. Because of the FIFO structure, all events (messages) have to be read until the message is returned that is propagated to the target application. The driver code for reading the receive FIFO is principally a while loop, and this can add the problem of non-deterministic latency times.

You resolve the latency time issue in the xPC Target CAN FIFO drivers by defining a receive FIFO read depth that is a constant number during application execution. For example, if you assume a FIFO read depth of 5, each time the Read Receive FIFO driver block is executed at the block sample time, the driver code reads and returns five events (messages) from the receive FIFO. This is independent of how many events the FIFO currently contains. There can be only two messages received in the FIFO and the third to fifth read attempt might just return the "No new event" code. Nevertheless, because the FIFO read latency does not exceed 17  $\mu$ s regardless of the event read out of the FIFO, the latency time becomes deterministic and is the Read FIFO Depth multiplied by 17  $\mu$ s. The driver block returns all new events and therefore all CAN messages going over the network. If only a small subset of the CAN messages received must be processed in the target application,

the total latency can easily exceed the latency encountered with the Dynamic Object Buffer mode for the same application. There is another restriction specific to the FIFO mode concept. Using more than one Read Receive FIFO block in a Simulink model is not recommended, because a new event (message) read by one block instance cannot be read again by another block instance (the event is no longer in the FIFO buffer). Therefore the entire CAN receive part has to be concentrated in one Read Receive FIFO block in your model. For the write transmit FIFO side, this restriction does not apply. Here you can use as many instances as you want.

The Setup block for the CAN FIFO mode controls the CAN acceptance filters of the CAN controller. The acceptance filter defines a range of CAN messages not to be forwarded to the receive FIFO. Filtering out unwanted CAN messages can drastically reduce the read receive FIFO latency time because the unwanted messages do not reach the receive FIFO. Unfortunately, the acceptance filter process uses binary evaluation, which does not allow filtering messages below and above a certain decimal range. Therefore the use of the acceptance filter only resolves the problem for a small subset of CAN network applications. See “Acceptance Filters” on page 5-6 for more information on this.

Look again at the example of 12 messages to be received and eight messages to be transmitted. If those 20 messages with their specific identifiers are the only messages going over the CAN network (100% usage ratio) the total latency time is

$$12 \cdot 17 \text{ s} + 8 \cdot 4 \text{ s} = 236 \text{ s}$$

This is a considerable smaller value than the 800  $\mu$ s that results when you use Dynamic Object Buffer mode drivers.

For the next case, assume that there are 12 additional messages going regularly over the network that do not need to be processed by the target application. Additionally, assume that those messages cannot be filtered by the CAN controller acceptance filter. Then the total latency time increases to

$$12 \cdot 17 \text{ s} + 20 \cdot 4 \text{ s} = 284 \text{ s}$$

There is no impact on the final result. That is the tradeoff. Therefore, the FIFO mode drivers are best suited for either CAN network monitoring applications or low-latency CAN applications where the ratio between the

number of messages to be processed and the number of total messages going over the network is high.

FIFO mode drivers are especially suited for monitoring type applications, because FIFO mode can return additional information such as the bus state or the reception of error frames. Dynamic Object Buffer mode drivers do not allow querying such information.

This documentation only covers the differences between the Dynamic Object Buffer mode drivers (standard drivers), and the FIFO mode drivers introduced here. It assumes that you are familiar with the Dynamic Object Buffer mode drivers and have successfully run one of the loop-back tests provided with xPC Target.

If you use FIFO mode drivers in your model, you must replace all Dynamic Object Buffer mode blocks (Setup, Send, Receive) with FIFO mode driver blocks. The CAN-AC2-xxx boards from Softing do not let you run the two CAN ports in different modes. Therefore the mode has to be same for both ports, but you can use more than one CAN board and run the boards in different modes just by selecting the correct I/O driver blocks.

As mentioned in the standard CAN chapter, you should not use the CAN-AC2 (ISA) for new projects. Instead use the CAN-AC2-PCI. Therefore, FIFO mode drivers are only provided for the CAN-AC2-PCI and the CAN-AC2-104 boards.

## Acceptance Filters

As mentioned earlier, you can use the CAN controller's acceptance filters to ensure that certain received messages referenced by their identifiers are written into the receive FIFO. Therefore, fewer read attempts are necessary to get at the messages that are of importance for the target application.

The behavior of the acceptance filter is described for standard and extended identifier ranges individually (one for standard identifiers and one for extended identifiers). Each acceptance filter is defined by a mask parameter and a code parameter.

The mask parameter defines, for each bit of the identifier, whether the filtering process cares about this bit or not. A 0 means "don't care" and a 1 means "do care."

The code parameter then defines, for each bit of the identifier, that the filtering process cares about (defined by the mask parameter), what the bit value has to be (0 or 1).

For standard identifiers the mask parameter and code parameter must be both, in the range 0 to 2047. For extended identifiers the mask parameter and code parameter must be both, in the range 0 to  $2^{29}-1$ .

The filtering process evaluates the following binary expression:

```
and( xor( mask, identifier ), code )
```

If all bits of the resulting value are 0, the message with this identifier is accepted. If any bit is 1, the message is voided.

According to this description, acceptance filters work using binary evaluation, while most applications differentiate messages (identifiers) in a decimal or hexadecimal manner. As a consequence, it is possible to filter messages, whose identifiers are above a certain decimal number. The opposite (identifiers below a certain decimal number) cannot be achieved in a general way.

## Examples

The default values for the mask parameter and the code parameter in the FIFO setup driver block are both 0. These parameter values ensure (the above expression always evaluates to 0) that all incoming messages will reach the receive FIFO (no filtering takes place). All parameter values are defined using decimal numbers. You can use the MATLAB function `hex2dec` to define hexadecimal numbers in the dialog box entry.

Assume a CAN application where messages with the following identifiers (standard) are crossing the CAN network:

2-30, 48-122 (decimal)

Additionally, only incoming messages 4-29 must be processed by the target application. Ideally, all messages not having identifiers 4-29 would be filtered out, but the mask and code parameters do not allow this exact scheme. The closest you can achieve is filtering out all messages with identifiers above 31 by using value  $2047-31=2016$  for the mask parameter and value 0 for the code parameter. The messages with identifier 0, 1, 2, and 3 cannot be filtered out and are returned by the FIFO read driver block, even if they do not need to be processed by the target application.

## Examples

This section includes the following topics:

Example 1 (p. 5-8)	Loop-back test from CAN port 1 to CAN port 2
Example 2 (p. 5-11)	Filtering out events
Example 3 (p. 5-12)	Constructing CAN messages dynamically at runtime
Example 4 (p. 5-13)	Stopping the execution when a FIFO overflow occurs
Example 5 (p. 5-14)	Stopping the execution when a FIFO overflow occurs
Example 6 (p. 5-14)	Using CAN acceptance filters

### Example 1

Start with a simple model using the FIFO Setup block, FIFO Write block, FIFO Read block, and FIFO Read Filter block. The entire CAN network consists of a single physical connection between CAN port 1 and port 2 (loop-back configuration). Both CAN ports must be terminated properly.

The objective of the application is the following:

- Send a message with extended identifier 5100 and change data every millisecond on port 1.
- Send a message with standard identifier 112 and change data every even millisecond on port 1.
- Send a message with standard identifier 114 and change data every odd millisecond on port 1.
- Read three events every millisecond from the receive FIFO on port 2.
- Display the incoming data of the three messages separately.
- Do not use acceptance filtering (all messages are accepted).

---

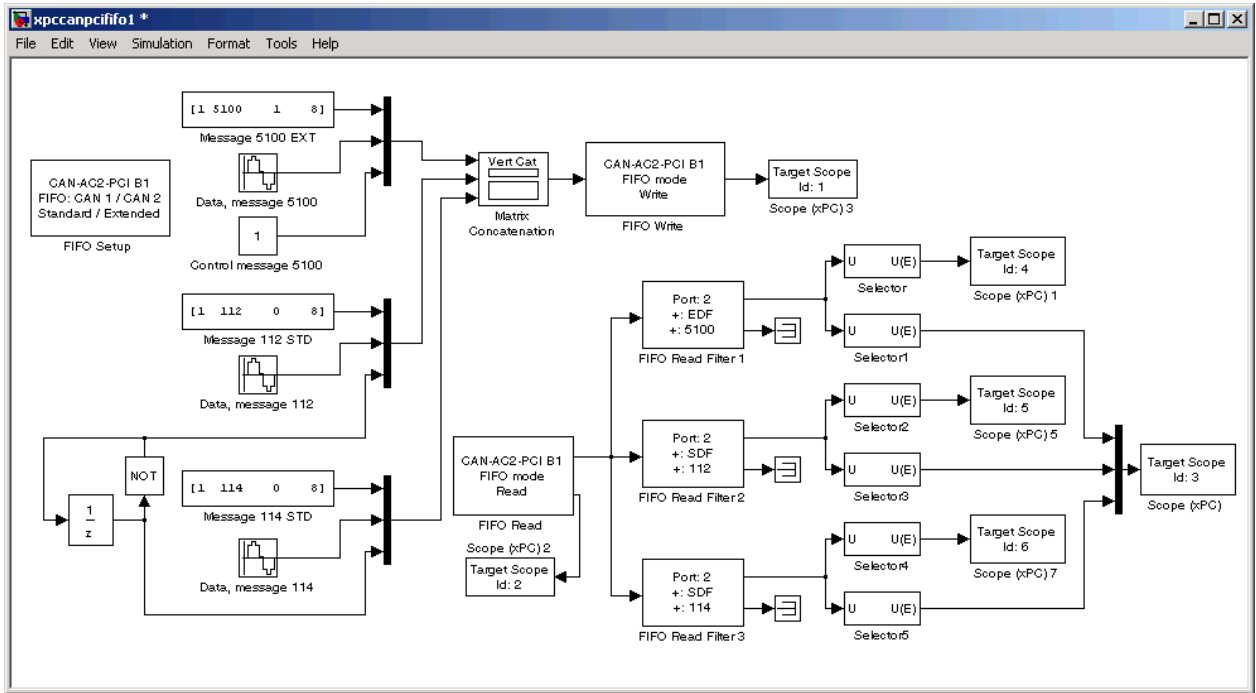
The data transmitted with the CAN messages are double values in all the following examples. This has been chosen for simplicity. You should refer to the bit-packing and bit-unpacking chapter of the standard CAN driver documentation on how to construct the data.

The first implementation uses the following scheme.

The matrix signal entering the FIFO Write block consists of all three messages including the Control element (sixth element); therefore, the matrix size is [3,6]. The sample time of the FIFO Write block is defined as 1 millisecond. For the standard identifiers that must be sent out every other millisecond, the Control element is alternated accordingly. You do this by using a Unit delay block with corresponding feedback as the Control element value.

The FIFO Read block has a read depth of 3 and also a base sample of 1 millisecond. Three FIFO filter blocks are connected to the output of the FIFO read block (in parallel) to extract the information of the incoming CAN messages.

You can display the model by typing, in the MATLAB Command Window, either `xpccanpcififo1` or `xpccan104fifo1`.



The model uses several xPC Target scope blocks to display different types of data on the target screen:

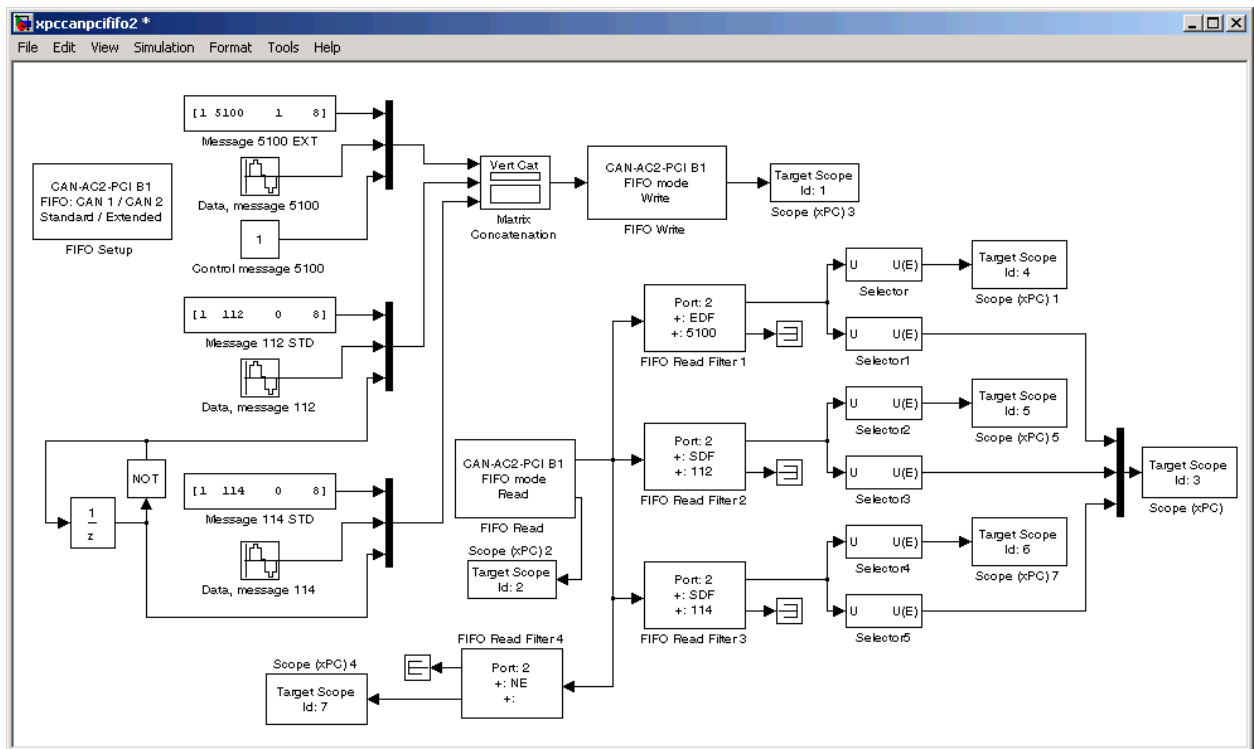
- Scope 1 (numerical) displays the status vector leaving the FIFO Write block.
- Scope 2 (numerical) displays the status vector [lost-message-counter, bus state] leaving the FIFO Read block.
- Scope 3 (graphical) plots the data of all three CAN messages being received.
- Scopes 4-6 (numerical) displays the other relevant message data of the three incoming CAN messages individually (port, identifier, type, data length, timestamp).



## Example 2

When looking at the time behavior of the model, you can observe that at each millisecond two CAN messages are sent out via the FIFO Write block, while the FIFO Read block reads three events each millisecond from the receive FIFO. This implies that one of the three events leaving the FIFO Read block is of type No new event. You can show this by attaching another FIFO Filter block in parallel that filters “No new events,” and then displaying the second output port, which reports the number of matching events.

You can display the model by typing, in the MATLAB Command Window, either `xpccanpcififo2` or `xpccan104fifo2`.



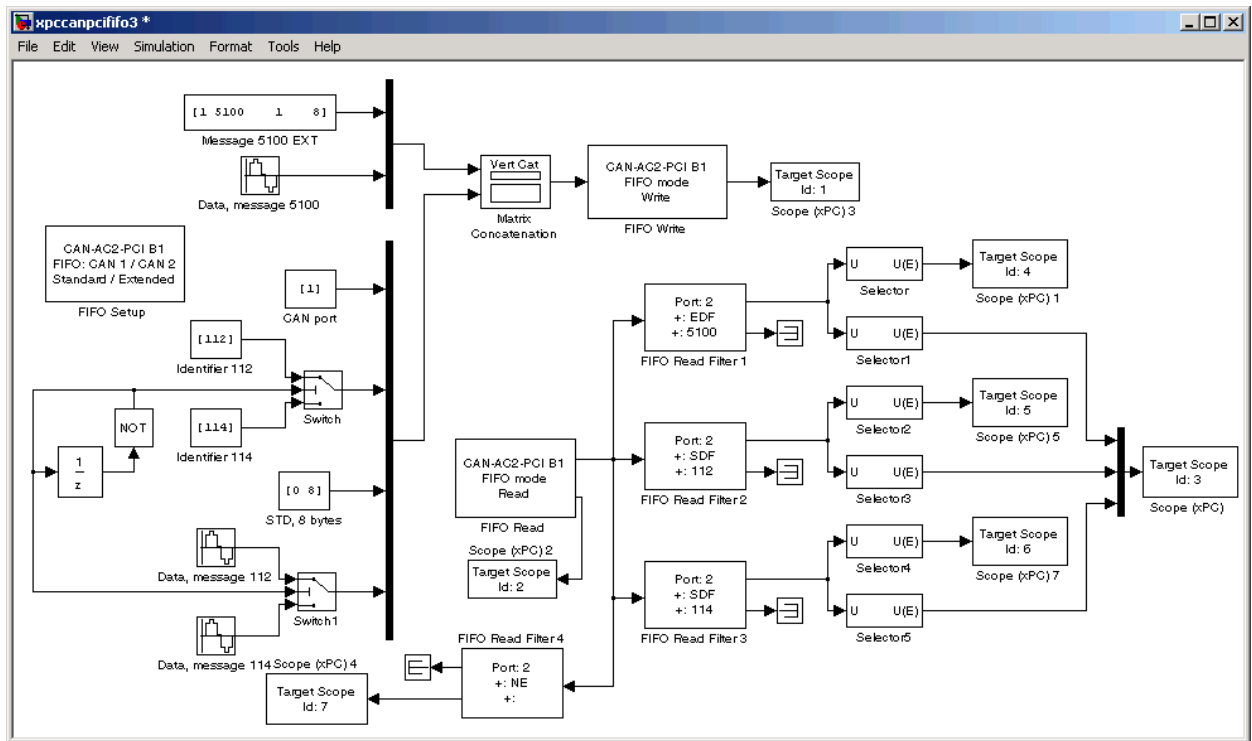
Having observed this, you can reduce the read depth of the FIFO Read block from 3 to 2. This would not change the overall behavior of the model. As a

positive side effect, the latency time of the FIFO Read block becomes smaller and therefore the model's cycle time as well.

### Example 3

Now look at a second implementation on the FIFO Write side. Instead of providing three messages in parallel, you can just write two messages and then alternate the identifier and data of the second CAN message to be sent. Because the messages are now sent every millisecond in any case, you need not provide the Control element, thereby reducing the matrix entering the FIFO Write block to a size of [2,5].

You can display the model by typing, in the MATLAB Command Window, either `xpccanpcififo3` or `xpccan104fiffo3`.

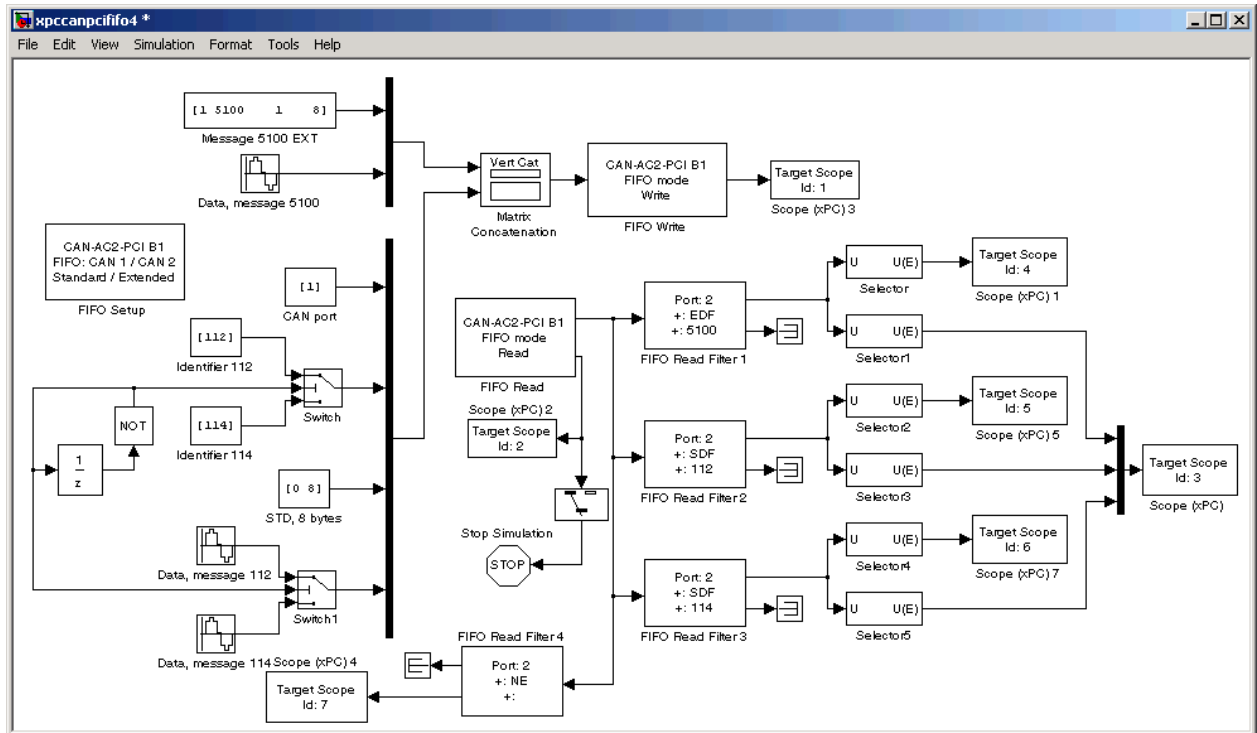


This implementation behaves exactly like the first implementation, but shows how CAN messages to be sent can be constructed dynamically at run-time.

## Example 4

Look at the situation where the read depth parameter of the FIFO Read block in the model above is set to 1 instead of 2 or 3. This leads to a receive FIFO overflow when the execution time reaches 256 milliseconds. Here, for example, the execution should be stopped if the overflow occurs. This can do this easily by evaluating the lost message counter value leaving the status output port of the FIFO Read block.

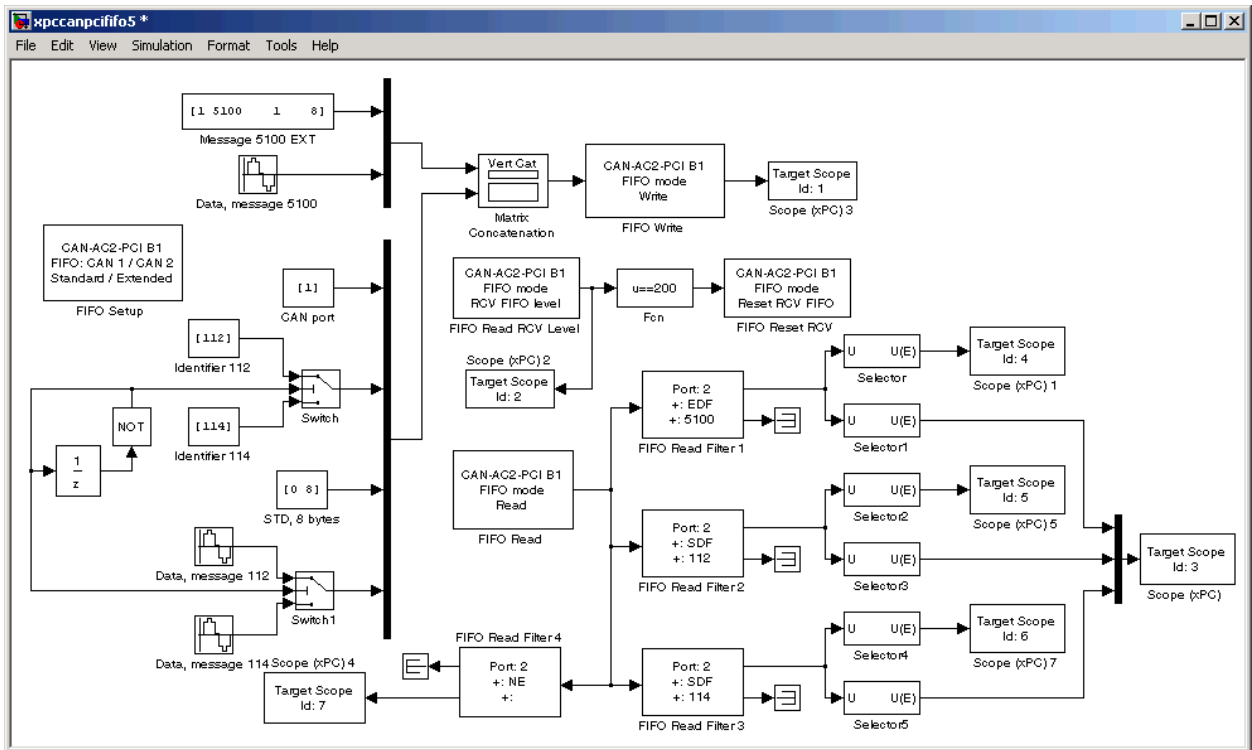
You can display the model by typing, in the MATLAB Command Window, either `xpccanpcififo4` or `xpccan104fifo4`.



### Example 5

Consider a different handling of the receive FIFO overflow: If the receive FIFO level reaches the value of 200, the receive FIFO should be reset to delete all currently stored events. The application execution has to continue normally. For this, you need to add two new driver blocks to the model, which are used to read the receive FIFO level and then reset it.

You can display the model by typing, in the MATLAB Command Window, either `xpccanpcififo5` or `xpccan104fifo5`.



### Example 6

The next example shows the use of the CAN acceptance filters. First the Read depth parameter of the FIFO Read block is set to a value of 2. Then the identifier of the second standard message is changed from 114 to 188. The

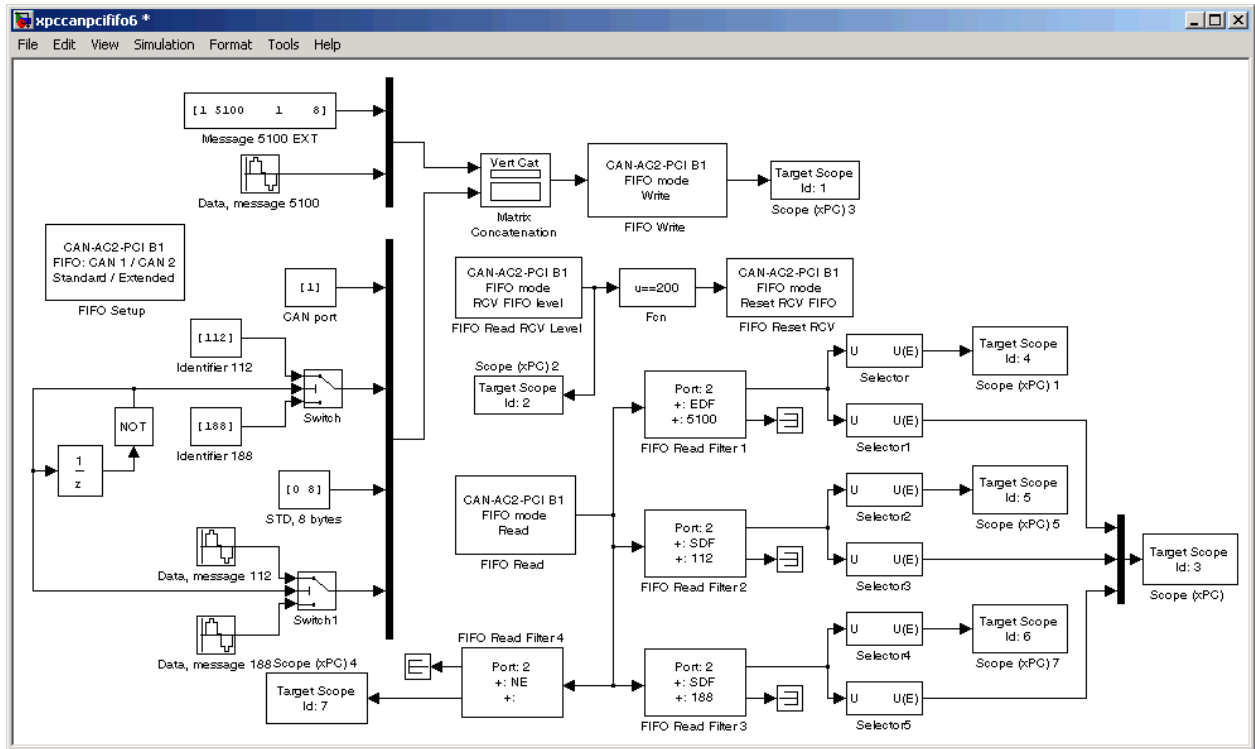
goal is to filter any CAN messages with an identifier larger than 127. This means that the receive FIFO never contain the CAN message with identifier 188. Also, the FIFO Filter block, filtering CAN message with identifier 114, is changed to filter the message with identifier 188. To achieve this, the **Acceptance Filters** parameter of CAN port 2 in the FIFO Setup block must be set accordingly:

```
[2047-127, 0, 0, 0]
```

If you rebuild and reexecute the target application, you can see the following:

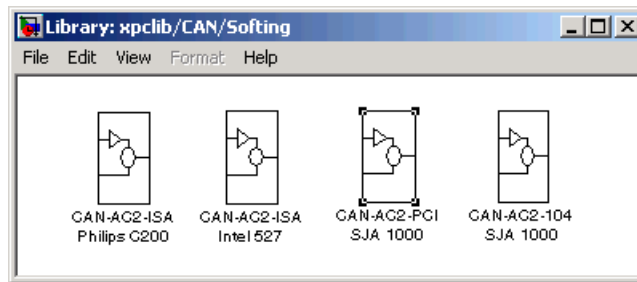
- Scope with ID 6 shows 0 for all elements of the vector leaving the corresponding FIFO Filter block. The message with identifier 188 is never received.
- Scope with ID 3 shows one of the data traces always being zero.
- Scope with ID 7 shows a value of 1, which reflects that the read depth could be reduced to 1, because only one message per millisecond reaches the receive FIFO now.

You can display the model by typing, in the MATLAB Command Window, either `xpccanpcififo6` or `xpccan104fiffo6`.

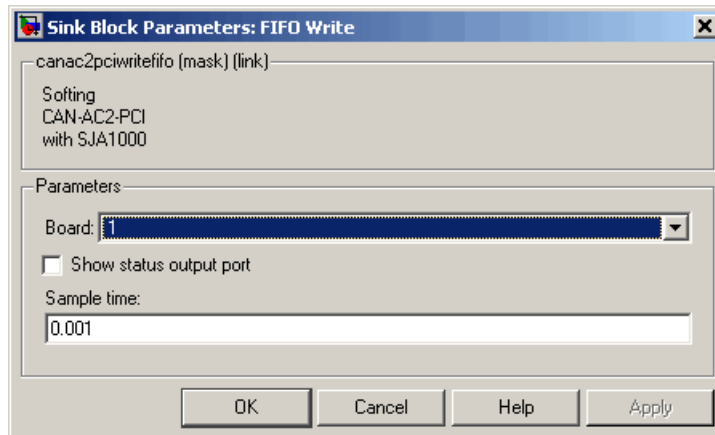


## CAN FIFO Blocks for the CAN-AC2-PCI with Philips SJA1000 CAN Controller

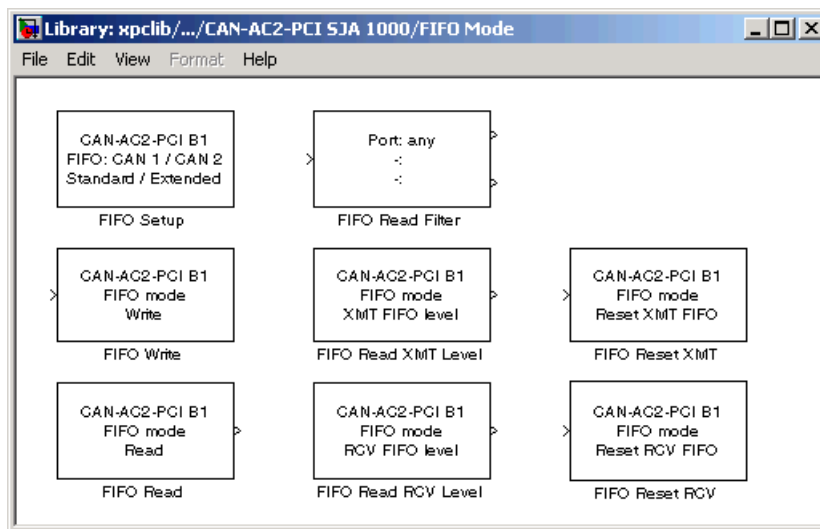
The driver blocks described here support the CAN-AC2-PCI using FIFO mode. The Philips SJA1000 chip is used as the CAN controller in this configuration and supports both standard and extended identifier ranges in parallel. The driver block set for this board is found in the xPC Target I/O block library in the group CAN/Softing.



The third block group, CAN-AC2-PCI SJA 1000, contains the FIFO mode subgroup.



The highlighted group contains all driver blocks available for FIFO Mode CAN.



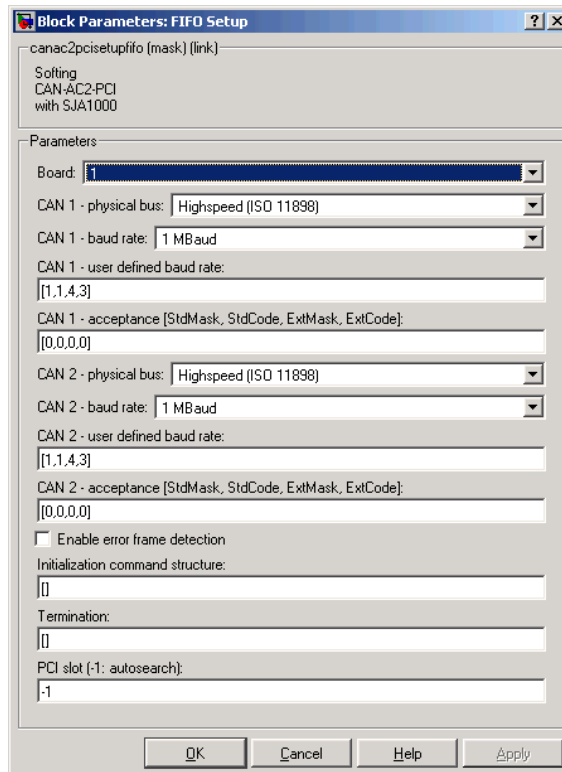


# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup

**Purpose** Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup block

**Library** xPC Target Library for CAN

**Description** The Setup block defines general settings of the installed CAN board(s). The CAN driver blocks for this board support up to three boards for each target system, making up to six CAN ports available. For each board in the target system, you can use one Setup block in a model.



## Block Parameters

### Board

Defines the board being accessed by this driver block instance. If multiple boards are present in the target PC, you can use the board number (1...3) to differentiate the boards. The physical

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup

---

board referenced by the board number depends on the **PCI Slot** parameter. If just one board is present in the target system, select board number 1.

## **CAN 1 - physical bus**

Defines the physical CAN bus type of CAN port 1. In the board's standard hardware configuration, only high-speed CAN is supported. By extending the board with low-speed CAN piggyback modules, you can also select low-speed CAN as the physical bus. Do not change this value to low-speed if no module is present for the corresponding CAN port. If the module is present (see the Softing user manual for how to install the modules), you can select between high-speed and low-speed CAN here.

## **CAN 1 - baud rate**

Defines the most common baud rates for CAN port 1. If special timing is necessary (baud rate), select User defined.

## **CAN 1 - user defined baud rate**

If you select User defined from the CAN 1 Baud rate list, enter the 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 the CAN 1 port. 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 in which 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.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup

---

For information on how to define the acceptance information to filter certain messages, see “Acceptance Filters” on page 5-6.

## **CAN 2 - physical bus**

Defines the physical CAN bus type of CAN port 2. In the board’s standard hardware configuration, only high-speed CAN is supported. By extending the board with low-speed CAN piggyback modules, you can also select low-speed CAN as the physical bus. Do not set this value should to low-speed if no module is present for the corresponding CAN port. If the module is present (see the Softing user manual on how to install the modules), you can select between high-speed and low-speed CAN here.

## **CAN 2- baud rate**

Defines the most common baud rates for CAN port 2. If special timing is necessary (baud rate), select User defined.

## **CAN 2 - user defined baud rate**

If you select User defined from the CAN 2 baud rate list, enter the 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 the CAN 2 port. 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 in which 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.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup

---

For information on how to define the acceptance information to filter certain messages, see “Acceptance Filters” on page 5-6.

## **Enable error frame detection**

If the CAN controller should detect error frames and forward these to the Receive FIFO, select this box. Selecting 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, selecting this box might increase the FIFO Read driver block latency time because the receive FIFO gets filled with additional events.

## **Initialization (struct) and Termination (struct)**

Define the CAN messages sent during initialization and termination of the Setup block. For more information, see the standard CAN driver documentation in “Defining Initialization and Termination CAN Messages” on page 4-11.

## **PCI Slot (-1: autosearch)**

Defines the PCI slot in which the referenced board (board number) resides. If only one CAN board is present in the target system, the value for this control should be -1 for autosearch. This value ensures that the xPC Target kernel automatically finds the board regardless of the PCI slot it is plugged into. If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. Use the xPC Target function `getxpcpci` to query the target system for installed PCI boards and the PCI slots they are plugged into. For more information see `help getxpcpci`.

The board allows you to terminate each of the two CAN ports separately by means of DIP switches at the rear panel. Refer to the Softing user manual on how to set the DIP switches. Both CAN ports must be terminated properly if you use the loop-back model provided to test the board and drivers.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write

---

<b>Purpose</b>	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write Driver block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	<p>The FIFO Write driver block is used to write CAN messages into the transmit FIFO. The firmware running in FIFO mode processes the information found in the transmit FIFO and finally puts the constructed CAN messages onto the bus.</p> <p>The block has one input port of type double. At this port, all necessary information has to be provided to construct valid CAN messages to be written into the transmit FIFO. For each CAN message, five elements are passed:</p> <ul style="list-style-type: none"><li>Port</li><li>Identifier</li><li>Identifier type</li><li>Data frame size</li><li>Data</li></ul> <p><b>Port</b> The value can be either 1 (port 1) or 2 (port 2) and defines the port the CAN message is sent from.</p> <p><b>Identifier</b> Identifier of the CAN message to be sent. If it is a standard CAN message the valid range is 0 to 2047. If the CAN message is extended, the range is 0 to <math>2^{29}-1</math>.</p> <p><b>Identifier type</b> The value can be either 0 (standard identifier range) or 1 (extended identifier range) and defines the identifier type of the outgoing CAN message.</p> <p><b>Data frame size</b> The value can be in the range of 0 to 8 and defines the data frame size in bytes of the outgoing CAN message.</p>

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write

---

## **Data**

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

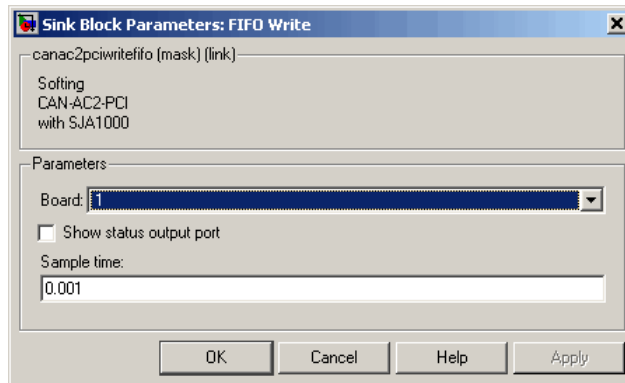
Because all this information can be dynamically changed in FIFO mode during application execution, the information is provided at the block input instead of you setting them through block parameters. To be able to transmit more than one CAN message per block instance, a matrix signal is used as a container for all information.

The dimension of the matrix signal entering the block has to 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 on how to construct the correct matrix signal for the FIFO write block, see “Examples” on page 5-8.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write

For certain applications it might be necessary to make the writing of a CAN message into the transmit FIFO dependent on the model dynamics. For this case, 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 into the transmit FIFO (1) or not (0).



## Block Parameters

### Board

Define the board used 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, you should select 1.

### Show status output port

Select this box to enable the status output port. If the box is cleared, 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 in which the number of elements depends on the signal dimension of the block input port. There is one element for each CAN message written into 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.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write

---

## **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 send CAN messages at different sample times. Or you can use multiple instances to structure your model more efficiently.



# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read

<b>Purpose</b>	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	<p>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 which the FIFO Read driver reads it.</p> <p>The FIFO Read driver block has at least one output port of type double. The signal of this port is a matrix of size <math>m \times 6</math>, where <math>m</math> is the FIFO Read depth defined in the block dialog box (see below). For example, if the FIFO read depth is 5, then the matrix signal of port 1 has size <math>5 \times 6</math>. Therefore, one row for each event is read out of the receive FIFO (no new message is considered an event as well). For information on how to extract data from the matrix signal, see “Examples” on page 5-8.</p> <p>Each row with its six elements contains all the information defining a CAN message:</p> <ul style="list-style-type: none"><li>Port</li><li>Identifier</li><li>Event type</li><li>Data frame size</li><li>Timestamp</li><li>Data</li></ul> <p><b>Port</b></p> <p>The value can be either 1 (port 1) or 2 (port 2) and reports the port at which the CAN message was received.</p> <p><b>Identifier</b></p> <p>Identifier of the CAN message being received. If it is a standard CAN message the range is 0 to 2047. It is an extended CAN message, the range is 0 to <math>2^{29}-1</math>.</p> <p><b>Event type</b></p> <p>This value defines the type of event read from the receive FIFO. The following values are defined in the Softing user manual.</p>

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read

---

- 0 No new event
- 1 Standard data frame received
- 2 Standard remote frame received
- 3 Transmission of a standard data frame is confirmed
- 4 -
- 5 Change of bus state
- 6 -
- 7 -
- 8 Transmission of a standard remote frame is confirmed
- 9 Extended data frame received
- 10 Transmission of an extended data frame is confirmed
- 11 Transmission of an extended remote frame is confirmed
- 12 Extended remote frame received
- 13 -
- 14 -
- 15 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.

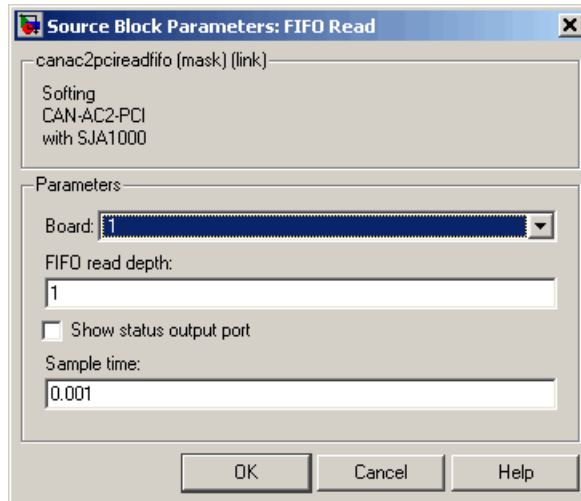
## **Timestamp**

Time at which the event was received. The resolution of the timestamp counter is 1  $\mu$ s.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read

## Data

Data of the data frame itself returned as a double value (8 bytes). The CAN Unpacking block is used to extract the data from the double value.



## Block Parameters

### Board

Defines the board used 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 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 number of events (CAN messages), which lead to a deterministic time behavior regardless of the number of events currently stored in the receive FIFO. The Read depth (m) also defines 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).

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read

---

## **Show status output port**

Select this box to 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 count up. This is an indicator that events (messages) will be unavoidably lost. The second element returns the current bus state. Possible values are

```
0 Error active  
1 Error passive  
2 Bus off
```

## **Sample time**

Defines the sample time at which the FIFO Read block is executed during a model (target application) run.

It is strongly recommended that you use only one instance of this block per physical CAN board in your model. Otherwise, you might get the unwanted behavior that one instance would read events that must be processed by blocks connected to the other, second instance.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read Filter

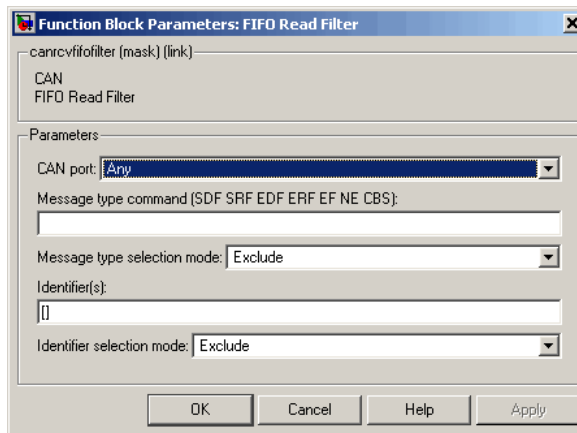
**Purpose** Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read Filter block

**Library** xPC Target Library for CAN

**Description** 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 and allows filtering events out of the event matrix, the signal leaving the FIFO Read driver block.

The block code walks through the rows of the incoming event matrix signal and looks for events matching the criteria defined in the block dialog box. If it matches, the entire event information (row) is written to the block's first output port. If more than one row matches the criteria, the later event overwrites the earlier event.

The block has one input port and two output ports. The input port is of type double and accepts a matrix signal of size  $m \times 6$ . The two output ports are of type double as well. The first output is a row vector ( $1 \times 6$ ), the filtered event, and the second outputs a scalar value that reports the number of matching events the filter block has processed.



# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read Filter

---

## Block Parameters

### CAN port

Defines the filter criterion for the CAN port. From the list, select 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 that are

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), from the **Message type command** parameter, 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, from the **Identifier(s)** parameter, 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 “Examples” on page 5-8.

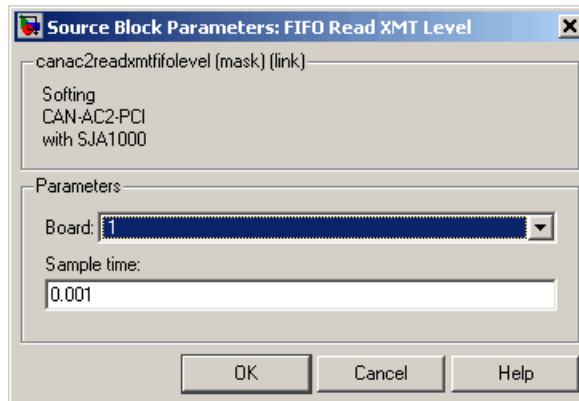
# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read XMT Level

**Purpose** Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read XMT Level block

**Library** xPC Target Library for CAN

**Description** 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 could stop the execution or wait for a non-full transmit FIFO.

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).



## Block Parameters

### Board

Defines the board accessed 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.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read XMT Level

---

## **Sample time**

Defines the sample time at which the FIFO Read XMT Level driver block is executed during a model (target application) run.



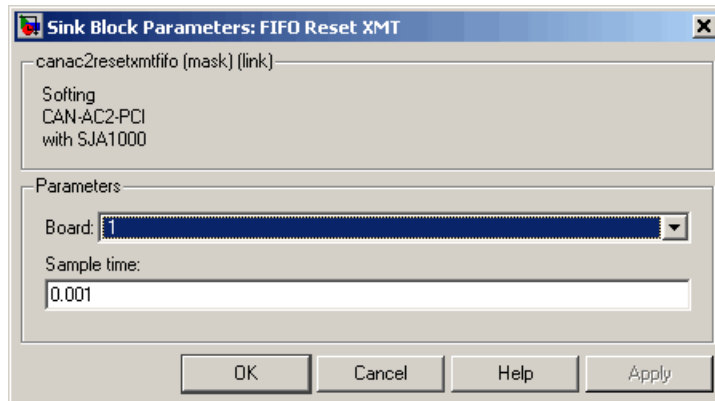
# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset XMT

**Purpose** Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset XMT block

**Library** xPC Target Library for CAN

**Description** The FIFO Reset XMT driver block is used to reset the transmit FIFOs. This deletes all messages currently stored in the transmit FIFO and reset the level counter to 0. As an example, you can use this driver block to reset the transmit 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 a scalar value of 0 is passed, no action takes place.



## Block Parameters

### Board

Defines the board accessed 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.

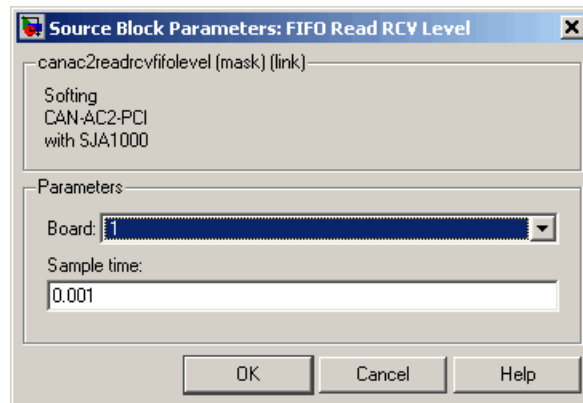
# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read RCV Level

**Purpose** Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read RCV Level block

**Library** xPC Target Library for CAN

**Description** The FIFO Read RCV level driver block reads 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, as shown by the lost message counter incrementing. You can use this driver block to check for this condition and take appropriate action, such as 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).



## Block Parameters

### Board

Defines the board accessed 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.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read RCV Level

---

## **Sample time**

Defines the sample time at which the FIFO Read RCV Level driver block is executed during a model (target application) run.

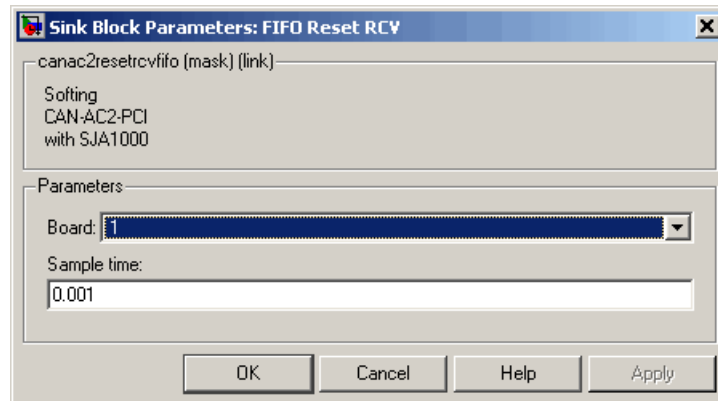
# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset RCV

**Purpose** Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset RCV block

**Library** xPC Target Library for CAN

**Description** The FIFO Reset RCV driver block resets the receive FIFO. This deletes all messages currently stored in the receive FIFO and reset 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 a scalar value of 0 is passed, no action takes place.



## Block Parameters

### Board

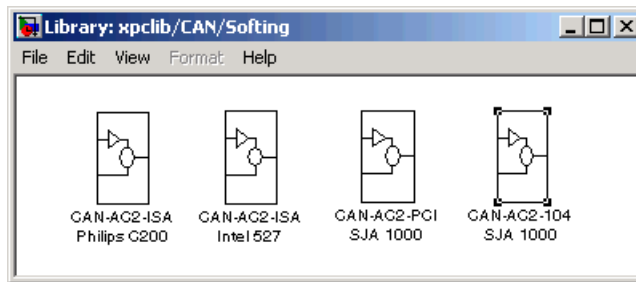
Defines the board accessed to reset the receive FIFO. For more information about the meaning of the board number, see the Setup driver block described above. If just one board is present in the target system, select board number 1.

### Sample time

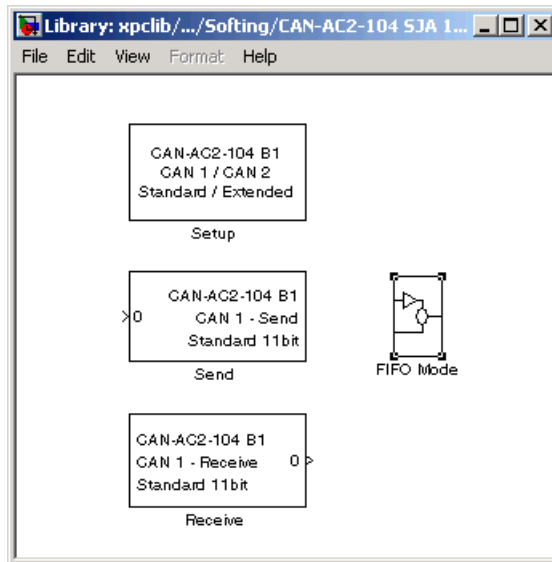
Defines the sample time at which the FIFO Reset RCV driver block is executed during a model (target application) run.

## CAN FIFO Blocks for the CAN-AC2-104 with Philips SJA1000 CAN Controller

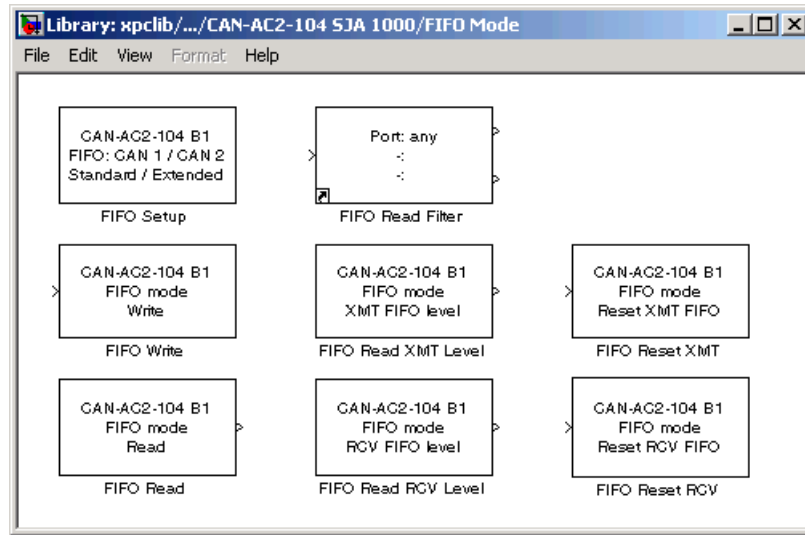
The driver blocks described here support the CAN-AC2-104 (PC/104) using FIFO mode. The Philips SJA1000 chip is used as the CAN controller in this configuration and supports both standard and extended identifier ranges in parallel. The driver block set for this board is found in the xPC Target I/O block library in the group CAN/Softing.



The fourth block group, CAN-AC2-104 SJA 1000, contains the FIFO Mode subgroup.



The highlighted group contains all the driver blocks available for FIFO mode CAN.



# Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup

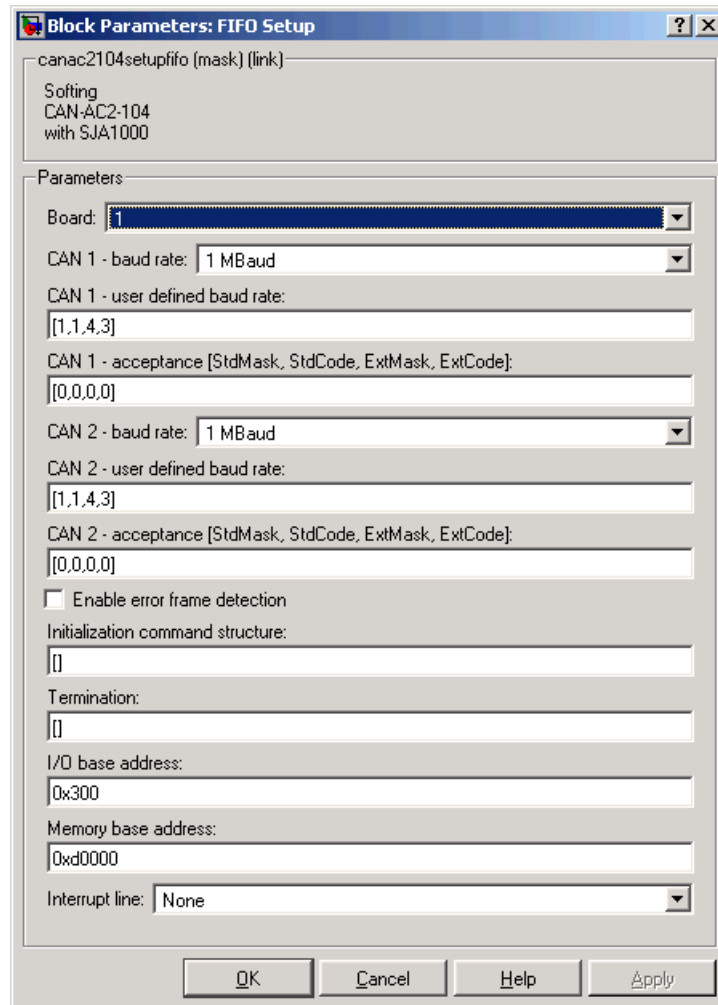
---

**Purpose** Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup block

**Library** xPC Target Library for CAN

**Description** The FIFO Setup driver block defines general settings of the installed CAN boards. The CAN driver blocks for this board support up to three boards for each target system, making up to six CAN ports available. For each board in the target system, you must use exactly one Setup driver block.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup



## Block Parameters

### Board

Defines the board accessed by this driver block instance. If multiple boards are present in the target PC, you can use the board number (1...3) to differentiate the boards. The physical



# Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup

board referenced by the board number depends on the **PCI Slot** parameter.

## **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 in which 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 "Acceptance Filters" on page 5-6.

## **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 selected **User defined** from the CAN 1 - baud rate list, enter four values for the timing information. The vector elements have the following meanings:

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup

---

[ 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 in which 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 “Acceptance Filters” on page 5-6.

## **Enable error frame detection**

Defines whether the CAN controller should detect error frames and forward these to the receive FIFO. Selecting 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, selecting 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 4, “CAN I/O Support”.

## **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 the memory base address the board should use. See the Softing user manual for this board on how

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup

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 4 KB. If more than one board is present in the target system, you must enter a different memory base address for each board and you must ensure that the defined address ranges do not overlap. Because the xPC Target kernel only reserves a subset of the address range between 64 KB and 1 MB for memory-mapped devices, the address ranges must be within the following range:

C0000-DC000

The board allows you to terminate 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 loop-back model provided to test the board and drivers.

## **Interrupt line**

Select an interrupt line from the list.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Write

---

**Purpose** Softing CAN-AC2-104 with Philips SJA1000 FIFO Write block

**Library** xPC Target Library for CAN

**Description** 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 constructed CAN messages onto the bus.

The block has one input port of type double. At this port, you must provide all necessary needed to construct valid CAN messages to be written into the transmit FIFO. For each CAN message, five elements have to be passed:

- Port
- Identifier
- Identifier type
- Data frame size
- Data

**Port**  
The value can be either 1 (port 1) or 2 (port 2) and defines at which port the CAN message is sent from.

**Identifier**  
Identifier of the CAN message to be sent. 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) and defines the identifier type of the outgoing CAN message.

**Data frame size**  
The value can be in the range of 0 to 8 and defines the data frame size in bytes of the outgoing CAN message

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Write

---

## Data

Data for the data frame itself, defined as a double value (8 bytes). The CAN Packing block is used to construct the data as a double value.

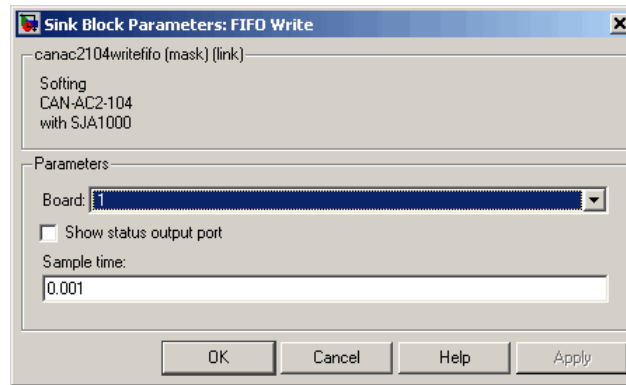
Because all this information can be dynamically changed in FIFO mode during application execution, the information is provided at the block input instead of using the block parameters. To transmit more than one CAN message per block instance, use a matrix signal as a container for all information.

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 “Examples” on page 5-8.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Write

For certain applications it might be necessary to make the writing of a CAN message into the transmit FIFO dependent 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 into the transmit FIFO (value 1) or not (value 0).



## Block Parameters

### Board

Defines the board used 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.

### 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 in which the number of elements depends on the signal dimension of the block input port. There is one element for each CAN message written into 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.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Write

---

## **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 with different sample times, you can send CAN messages out at different rates. Or you can use multiple instances to structure your model more efficiently.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read

---

**Purpose** Softing CAN-AC2-104 with Philips SJA1000 FIFO Read block

**Library** xPC Target Library for CAN

**Description** The FIFO Read driver block reads 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 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 the FIFO Read RCV Level driver block to check for this condition and take appropriate action (for example, like stopping the execution or resetting the receive FIFO).

The FIFO Read driver block has at least one output port of type double. The signal of this port is a matrix of size  $m \times 6$ , where  $m$  is the FIFO read depth defined in the block's dialog box (see below). For example, if the FIFO read depth is 5, the matrix signal of port 1 has size  $5 \times 6$ . Therefore, there is one row for each event read from the receive FIFO (no new message is considered as an event as well). For information on how to extract data from the matrix signal, see "Examples" on page 5-8.

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 the port at which the CAN message was received.



# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read

---

## 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

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.

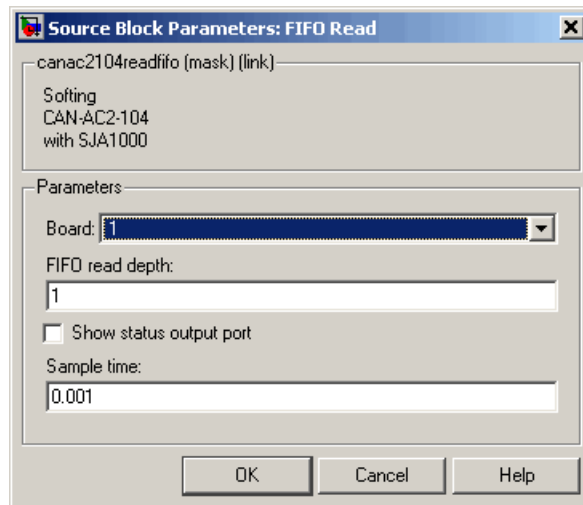
## Timestamp

Reports the time at which the event was received. The resolution of the timestamp counter is  $1\mu\text{s}$ .

## Data

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.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read



## Block Parameters

### Board

Defines the 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 number 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 ( $m$ ) also defines 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

Lets you enable the status output port. If the box is cleared (disabled), the block has one output port for events. If enabled, a second port is shown. The signal leaving that port is a vector of type double with two elements:

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read

---

[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.

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

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read Filter

---

**Purpose** Softing CAN-AC2-104 with Philips SJA1000 FIFO Read Filter Block

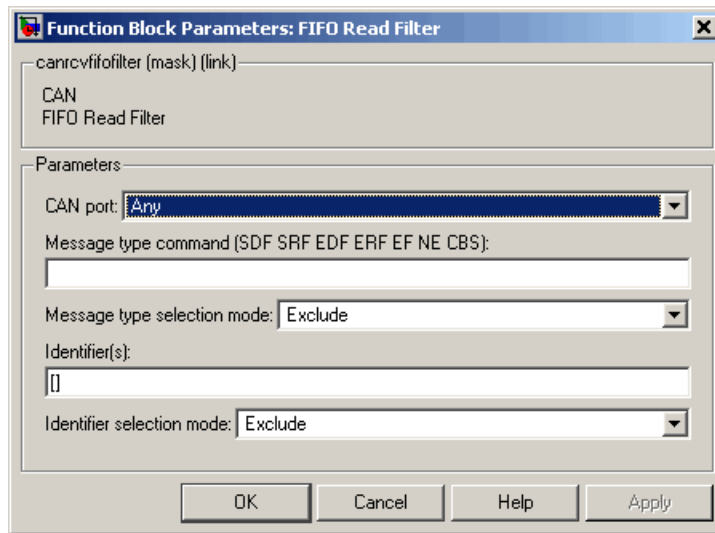
**Library** xPC Target Library for CAN

**Description** 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 match is found, the entire event information (row) is written to the block's first output port. If more than one row matches the criteria, the later event overwrites the earlier event.

The block has one input port and two output ports. The input port is of type double and accepts a matrix signal of size  $m \times 6$ . The two output ports are of type double as well. The first output is a row vector ( $1 \times 6$ ), the filtered event. The second output is a scalar value that reports the number of matching events the filter block has processed.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read Filter



## Block Parameters

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

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read Filter

---

## **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 **Identifier(s)** parameter 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 “Examples” on page 5-8.

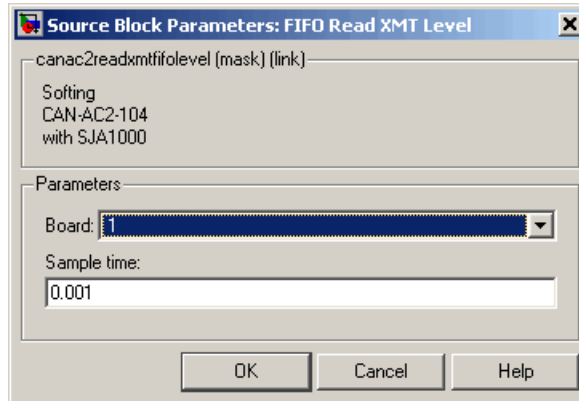
# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read XMT Level

**Purpose** Softing CAN-AC2-104 with Philips SJA1000 FIFO Read XMT Level block

**Library** xPC Target Library for CAN

**Description** 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).



## Block Parameters

### Board

Defines the 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.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read XMT Level

---

## **Sample time**

Defines the sample time at which the FIFO Read XMT Level driver block is executed during a model (target application) run.



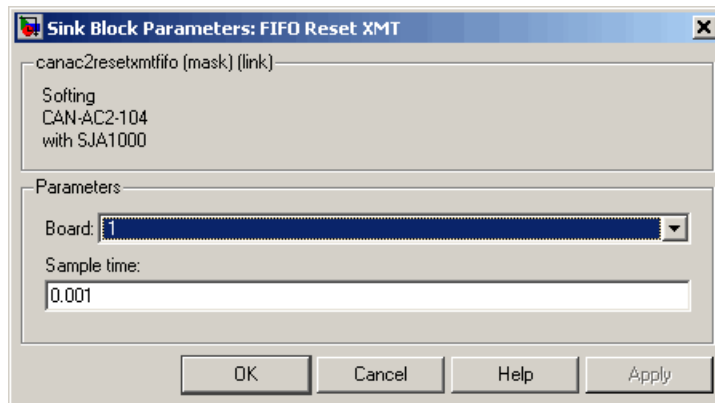
# Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset XMT

**Purpose** Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset XMT block

**Library** xPC Target Library for CAN

**Description** The FIFO Reset XMT driver block resets 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.



## Block Parameters

### Board

Defines the 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.

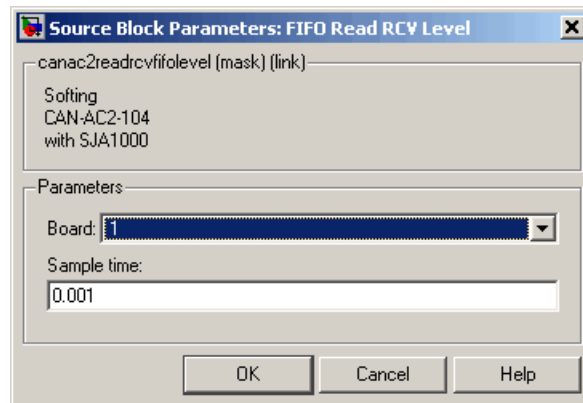
# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read RCV Level

**Purpose** Softing CAN-AC2-104 with Philips SJA1000 FIFO Read RCV Level block

**Library** xPC Target Library for CAN

**Description** The FIFO Read RCV Level driver block reads 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).



## Block Parameters

### Board

Defines the 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.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read RCV Level

---

## **Sample time**

Defines the sample time at which the FIFO Read RCV Level driver block is executed during a model (target application) run.

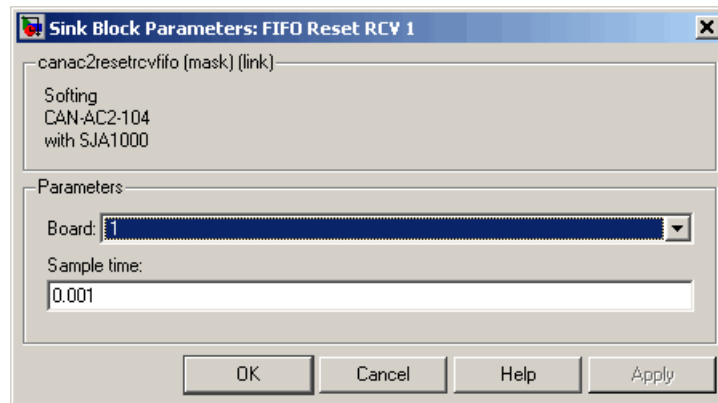
# Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset RCV

**Purpose** Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset RCV block

**Library** xPC Target Library for CAN

**Description** The FIFO Reset RCV driver block resets the receive FIFO. This deletes all messages currently stored in the receive FIFO and resets the level counter to 0. For example, you can use this driver block to reset the receive 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.



## Block Parameters

### Board

Defines the 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.

# UDP I/O Support

---

xPC Target provides support for the UDP/IP communication protocol. This chapter includes the following sections:

User Datagram Protocol (UDP) (p. 6-2)	Send and receive messages from a target application using UDP packets.
xPC Target UDP Example (p. 6-5)	Communicate between two xPC Target applications, between a target application and Simulink models or systems, or between two Simulink models.
UDP Communication Setup (p. 6-11)	Description of how to set up UDP communication
Boards and Blocks — Alphabetical List (p. 6-13)	Description of the block parameter fields for the xPC Target blocks that support UDP communication.

## User Datagram Protocol (UDP)

xPC Target supports communication from the target PC to other systems or devices using User Datagram Protocol (UDP) packets. UDP is a transport protocol similar to TCP. However, unlike TCP, UDP provides a direct method to send and receive packets over an IP network. UDP uses this direct method at the expense of reliability by limiting error checking and recovery. This section includes the following topics:

- “What Is UDP?” on page 6-2
- “Why UDP?” on page 6-4
- “Note on UDP Communication” on page 6-4

To use UDP for your xPC Target system, be sure to create a TCP/IP boot disk and boot the target PC with that boot disk.

### What Is UDP?

The User Datagram Protocol (UDP) is a transport protocol layered on top of the Internet Protocol (IP) and is commonly known as UDP/IP. It is analogous to TCP/IP. A convenient way to present the details of UDP/IP is by comparison to TCP/IP as presented below:

- Connection Versus Connectionless — TCP is a *connection based* protocol, while UDP is a *connectionless* protocol. In TCP, the two ends of the communication link must be connected at all times during the communication. An application using UDP prepares a packet and sends it to the receiver’s address without first checking to see if the receiver is ready to receive a packet. If the receiving end is not ready to receive a packet, the packet is lost
- Stream Versus Packet — TCP is a *stream-oriented* protocol, while UDP is a *packet-oriented* protocol. This means that TCP is considered to be a long stream of data that is transmitted from one end to the other with another long stream of data flowing in the other direction. The TCP/IP stack is responsible for breaking the stream of data into packets and sending those packets while the stack at the other end is responsible for reassembling the packets into a data stream using information in the packet headers. UDP, on the other hand, is a packet-oriented protocol

where the application itself divides the data into packets and sends them to the other end. The other end does not have to reassemble the data into a stream. Note, some applications might indeed present the data as a stream when the underlying protocol is UDP. However, this is the layering of an additional protocol on top of UDP, and it is not something inherent in the UDP protocol itself.

- **TCP Is a Reliable Protocol, While UDP Is Unreliable** — The packets that are sent by TCP contain a unique sequence number. The starting sequence number is communicated to the other side at the beginning of communication. Also, the receiver acknowledges each packet, and the acknowledgment contains the sequence number so that the sender knows which packet was acknowledged. This implies that any packets lost on the way can be retransmitted (the sender would know that they did not reach their destination because it had not received an acknowledgment). Also, packets that arrive out of sequence can be reassembled in the proper order by the receiver.

Further, time-outs can be established, because the sender will know (from the first few packets) how long it takes on average for a packet to be sent and its acknowledgment received. UDP, on the other hand, simply sends the packets and does not keep track of them. Thus, if packets arrive out of sequence, or are lost in transmission, the receiving end (or the sending end) has no way of knowing.

TCP communication can be compared to a telephone conversation where a connection is required at all times and two-way streaming data (the words spoken by each party to the conversation) are exchanged. UDP, on the other hand, can be compared to sending letters by mail (without a return address). If the other party is not found, or the letter is lost in transit, it is simply discarded. The analogy fails, however, when considering the speed of communication. Both TCP and UDP communication roughly happen at the same speed, because both use the underlying Internet Protocol (IP) layer.

---

**Note** *Unreliable* is used in the sense of “not guaranteed to succeed” as opposed to “fails a lot of the time.” In practice, UDP is quite reliable as long as the receiving socket is active and is processing data as quickly as it arrives.

---

## Why UDP?

UDP was chosen as the transport layer for xPC Target precisely because of its lightweight nature. Since the primary objective of an application running in the xPC Target framework is real-time, the lightweight nature of UDP ensures that the real-time application will have a maximum chance of succeeding in real-time execution. Also, the datagram nature of UDP is ideal for sending samples of data from the Real-Time Workshop-generated application. Because TCP is stream oriented, separators between sets of data must be used for the data to be processed in samples. It is easier to build an application to deal with unreliable data than it is to decode all of this information in real-time. If the application is unable to process the data as quickly as it arrives, the following packets can just be ignored and only the most recent packet can be used.

Communication can involve a packet made up of any Simulink data type (double, int8, int32, uint8, etc.), or a combination of these. xPC Target provides blocks for combining various signals into one packet (packing), and then transmitting it. xPC Target also provides blocks for splitting a packet (unpacking) into its component signals that can then be used in a Simulink model. The maximum size of a packet is limited to about 500 bytes.

## Note on UDP Communication

The UDP blocks work in the background when the real-time application is not running. The UDP communication has been set up to have a maximum of two UDP packets waiting to be read. This applies to one UDP port, which corresponds to one UDP Receive block. All subsequent packets are rejected. This prevents excessive memory usage and minimizes the load on the TCP/IP stack. Consequently, when any large background task is performed, such as uploading a screen shot or communicating large pages through the WWW interface, packet loss might occur. Design applications so that this is not critical. In other words, the receipt of further packets after the ones that were lost ensures seamless continuation.

Note also that because UDP block transfers operate as background tasks, xPC Target disables them in polling mode. See Restrictions Introduced by Polling Mode.



## xPC Target UDP Example

This section provides an example of how to set up two-way data exchange between two xPC Target systems, between xPC Target and Simulink, or between two Simulink models. When one or both of the systems is running Simulink in non-real time, be sure to set the sample time properly.

---

**Note** To use UDP for your xPC Target system, be sure to create a TCP/IP boot disk and boot the target PC with that boot disk.

---

The hypothetical models are called `udpsendreceiveA` and `udpsendreceiveB`. Two different sets of data are transferred between these two models, one set from `udpsendreceiveA` to `udpsendreceiveB` and another set in the opposite direction.

The data to transfer is in the following order:

`udpsendreceiveA` to `udpsendreceiveB`

- `uint8` (3x3)
- `uint16` (1x1)
- `double` (2x4)

`udpsendreceiveB` to `udpsendreceiveA`

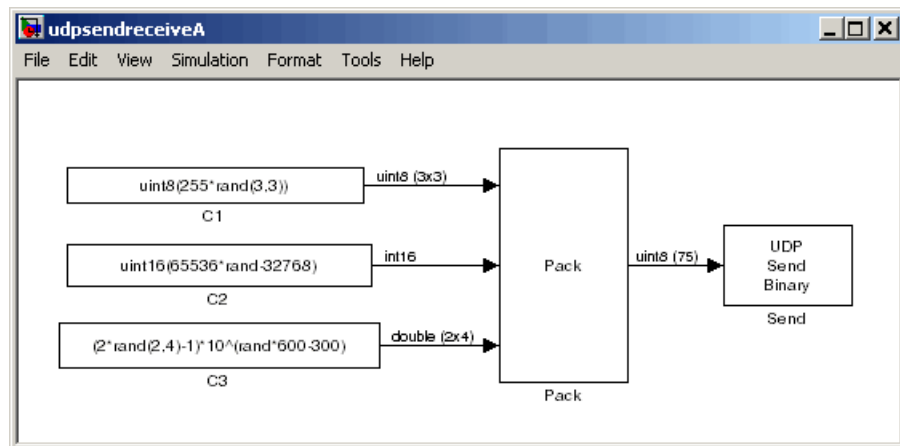
- `single` (4x1)
- `double` (2x2)
- `uint32` (2x2)
- `int8` (5x3)

For the purposes of this example, all the inputs are generated using Simulink Constant blocks that use the MATLAB random number function (`rand`). The random numbers are generated by Real-Time Workshop using this function at the time of code generation. To generate the vector of `uint8` (3x3), use the MATLAB function

```
uint8(255 * rand(3,3))
```

since 255 is the maximum value for an unsigned 8-bit integer. The other values are generated similarly.

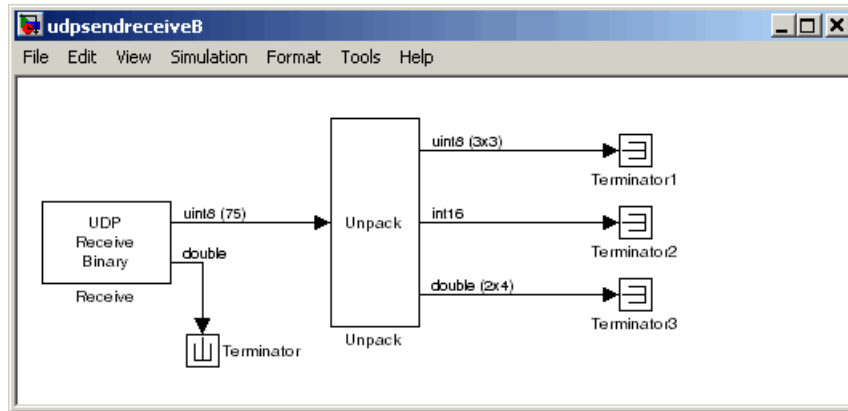
With this setup, construct the send side of `udpsendreceiveA`.



Note that the width of the UDP packet to be sent is 75 bytes. The parameters used in the Pack block are **Input port datatypes** `{'uint8', 'uint16', 'double'}` and **Byte Alignment** 1.

For the Send block, set the **IP Address to send to** to `192.168.0.2`. This is the hypothetical address of the system that will run `udpsendreceiveA`. Set the **IP Port to send to** to 25000 (picked arbitrarily). The sample time is set to 0.01.

Use this information to construct the receive end of udpsendreceiveB.

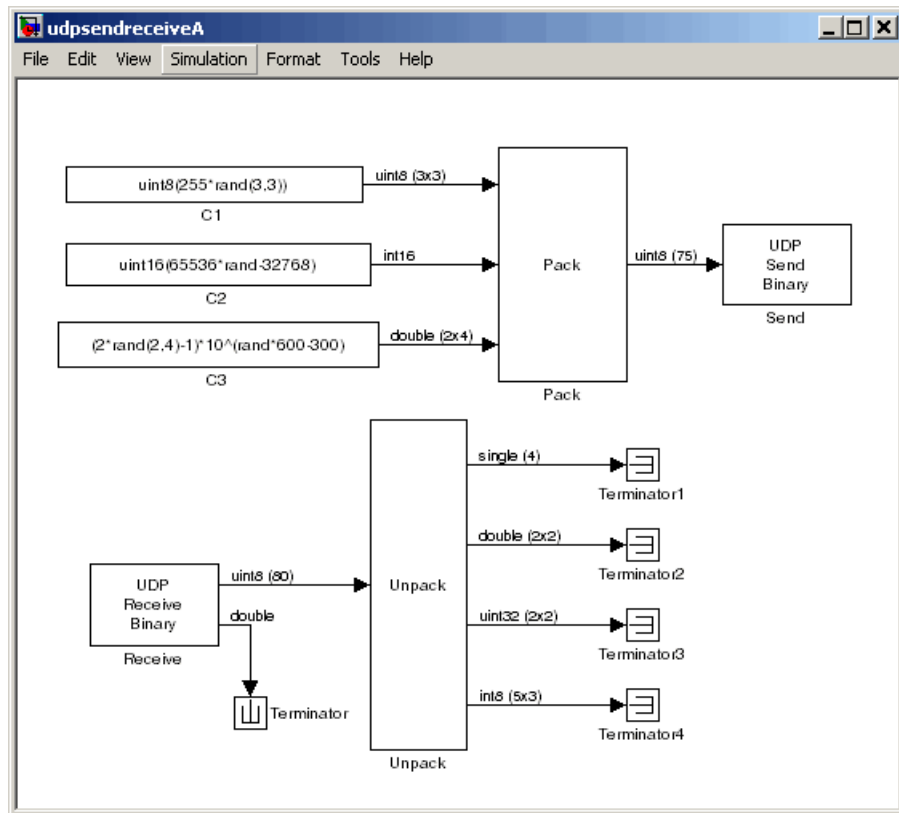


For setting up the Receive block, set **IP address to receive from** to 192.168.0.2 (the hypothetical address of the system that will run udpsendreceiveB). The **IP port to receive from** is set to 25000 (the same value as set in the Send block in udpsendreceiveA). The **Output port width** is set to 75, which is obtained from the output port width of the Pack block in udpsendreceiveA.

For the Unpack block, **Byte Alignment** is set to 1 and the **Output port datatypes** is set to {'uint8', 'int16', 'double'} from the Pack block in udpsendreceiveA. The **Output port dimensions** is set to {[3 3], 1, [2 4]} from the dimensions of the inputs to the Pack block in udpsendreceiveA.

Note that in udpsendreceiveB, the second output port of the Receive block is sent into a terminator. You can use this to determine when a packet has arrived. The same is true for the outputs of the Unpack block, which in a real model would be used in the model.

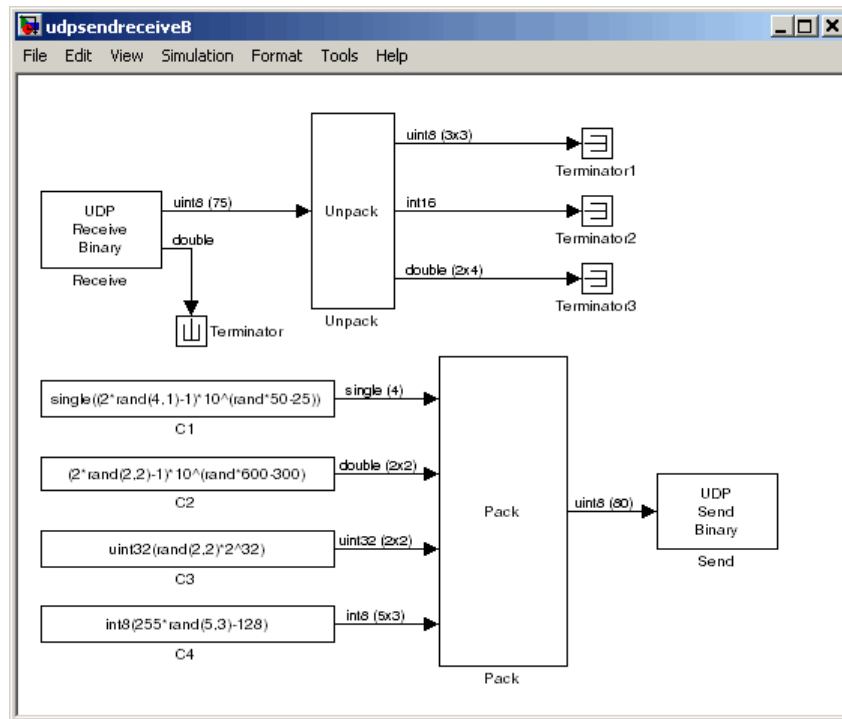
To construct the udpsendreceiveB to udpsendreceiveA side of the communication, follow an analogous procedure. The final udpsendreceiveA is shown below.



The following table lists the parameters in `udpreceiveA`.

Block	Parameter	Value
Receive	IP address	192.168.0.1
	IP port	25000
	Output port width	80
	Sample time	0.01
Unpack	Output port dimensions	{4,[2 2],[2 2],[5 3]}
	Output port data types	{'single','double','uint32','int8'}
	Byte alignment	2

The final `udpreceiveB` model is shown below.



The following table lists the parameters in `udpreceiveB`.

<b>Block</b>	<b>Parameter</b>	<b>Value</b>
Pack	Input port data types	{ 'single', 'double', 'uint32', 'int8' }
	Byte alignment	2
Send	IP address	192.168.0.1
	IP port	25000
	Sample time	0.01

## UDP Communication Setup

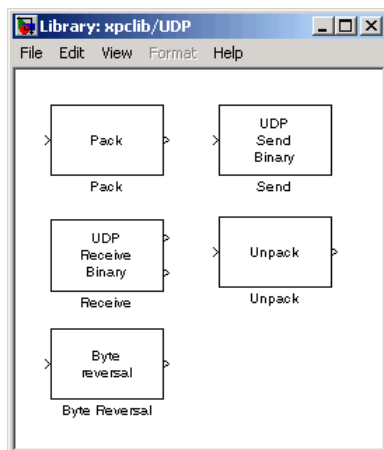
The infrastructure provided in the xPC Target Library for UDP communication consists mainly of two blocks — a Send block and a Receive block. These blocks are in the xPC Target Library, available from the Simulink Library under **xPC Target**. You can also access them from the MATLAB command line by typing

```
xpclib
```

The blocks are located under the UDP heading in the library. The Send block takes as input a vector of type `uint8`, which it sends. This is limited to a length of about 500 bytes (i.e., a 1 by 500 vector). Similarly, the Receive block outputs a vector of `uint8`. To convert arbitrary Simulink data types into this vector of `uint8`, a Pack block is provided, while an Unpack block is provided to convert a vector of `uint8`s back into arbitrary Simulink data types.

You can have up to 32 UDP blocks in any given model (Send and Receive blocks combined in any order).

xPC Target includes a Byte Reversal block for communication with *big-endian* architecture systems. You do not need this block if you are communicating between 80x86-based PC systems running either xPC Target or Microsoft Windows.



All the blocks are set up to work both from within Simulink and from an application running under xPC Target. However, you must be careful when using a Simulink simulation and an xPC Target application to communicate, or when using two Simulink models. This is because a Simulink model executes in non-real time and can be several times faster or slower than real time. The sample time of the send and receive blocks and the sample time of the Simulink model must be set so that the communication can proceed properly.

Note the following:

- Your model cannot have two UDP Receive blocks configured with the same receive port, for example, the same receive port and different IP addresses.
- Your model cannot have two UDP Send blocks configured with the same send (not -1) port, for example, the send port and different IP addresses.



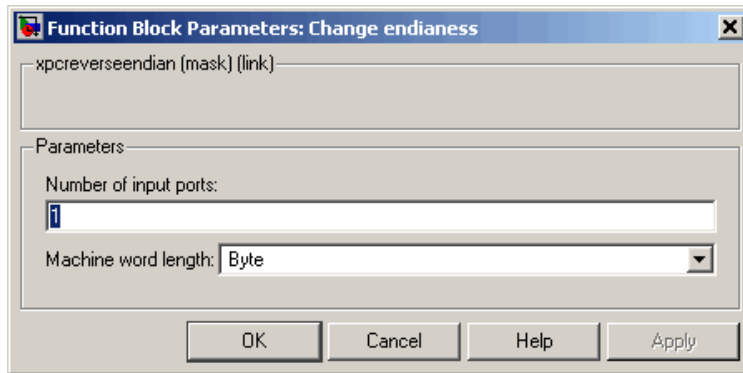
## **Boards and Blocks – Alphabetical List**

# Byte Reversal/Change Endianess

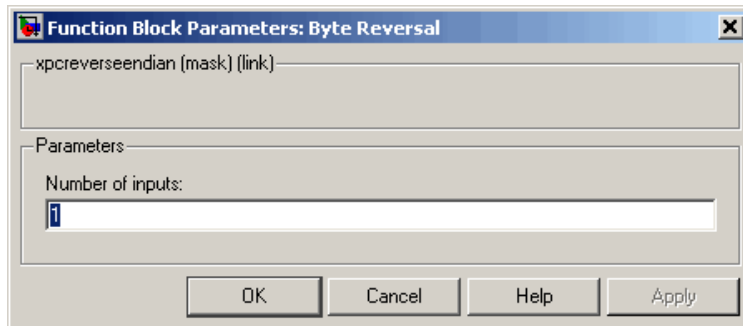
---

**Purpose** Byte Reversal/Change Endianess block

**Description** You use the Byte Reversal/Change Endianess block for communication between an xPC Target system and a system running with a processor that is *big-endian*. Processors compatible with the Intel 80x86 family are always *little-endian*. For this situation, you should insert a Byte Reversal/Change Endianess block before the Pack block and just after the Unpack block to ensure that the values are transmitted properly. The following is the Change Endianess block.



The following is the Byte Reversal block.



## **Block Parameters for Change Endianess**

### **Number of input ports**

The number of input ports adjusts automatically to follow this parameter, and the number of outputs is equal to the number of inputs.

### **Machine word length**

Select one of the following machine word lengths to which to convert the data:

- Byte
- Word
- Double Word

The following is the Byte Reversal block.

## **Byte Reversal Block Parameters**

### **Number of inputs**

The number of input ports adjusts automatically to follow this parameter, and the number of outputs is equal to the number of inputs.

# UDP Pack

---

**Purpose** UDP Pack block

**Library** xPC Target Library for UDP

**Description** The Pack block is used to convert one or more Simulink signals of varying data types to a single vector of uint8 as required by the Send block.

## Block Parameters

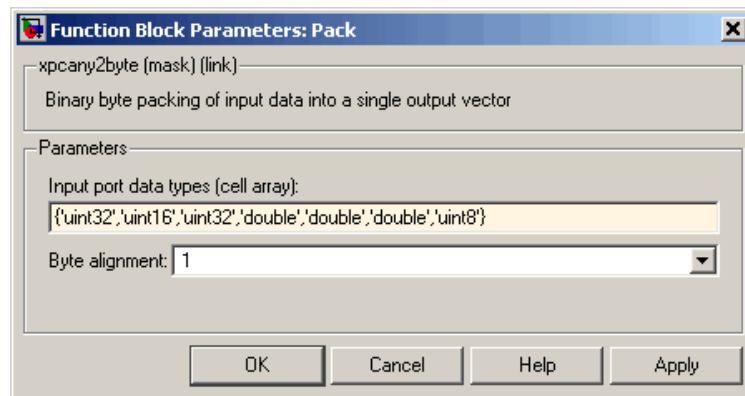
### Input port data types

Specify the data types for the different signals as part of the block parameters. The supported data types are double, single, int8, uint8, int16, uint16, int32, uint32, and boolean. The block determines the sizes of the signals automatically.

### Byte alignment

The byte alignment field specifies how the data types are aligned. The possible values are: 1, 2, 4, and 8. The byte alignment scheme is simple, and ensures that each element in the list of signals starts on a boundary specified by the alignment relative to the start of the vector.

## Example



As seen in the figure above, the data types of each of the signals must be specified as a cell array of strings in the correct order. Once this is done, the block automatically converts itself to one with the correct

number of input ports. There is always one output port. For example, say the Input port data types are specified as

```
{'uint8','uint32','single','int16','double'}
```

and an alignment of 4 is used. Assume also that all the signals are scalars. The first signal then starts at byte 0 (this is always true), the second at byte 4, the third at byte 8, the fourth at byte 12, and the fifth at byte 16. Note that the sizes of the data types used in this example are 1, 4, 4, 2, and 8 bytes respectively. This implies that there are “holes” of 3 bytes between the first and second signal and 2 bytes between the fourth and fifth signal.

A byte alignment of 1 means the tightest possible packing. That is, there are no holes for any combination of signals and data types.

---

**Note** Individual elements of vector/matrix signals are not byte aligned: only the entire vector/matrix is byte aligned. The individual elements are tightly packed with respect to the first element.

---

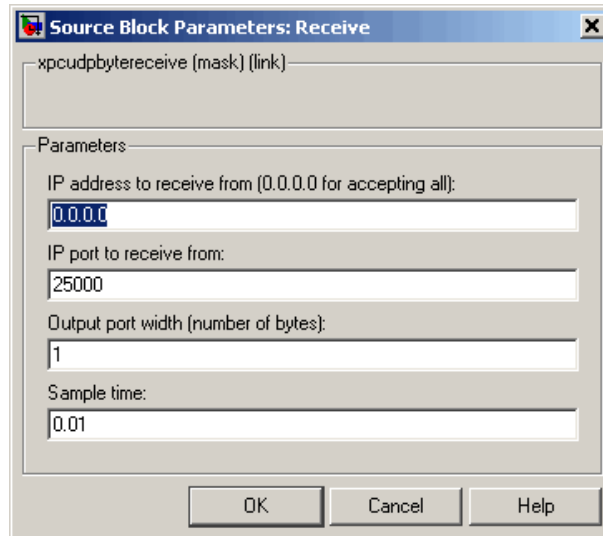
# UDP Receive

---

**Purpose** UDP Receive block

**Library** xPC Target Library for UDP

**Description** The Receive block has two output ports. The first port is the output of the received data as a vector of `uint8` while the second one is a flag indicating whether any new data has been received. This port outputs a value of 1 for the sample when there is new data and a 0 otherwise. The default behavior of the Receive block is to keep the previous output when there is no new data. You can modify this behavior by using the second port to flag when there is new data.



## Block Parameters

### IP address to receive from

Can be left with the default value of 0.0.0.0. This accepts all UDP packets from any other computer. If set to a specific IP address, only packets arriving from that IP address are received.

**IP port to receive from**

Port that the block accepts data from. The other end of the communication sends data to the port specified here.

**Output port width**

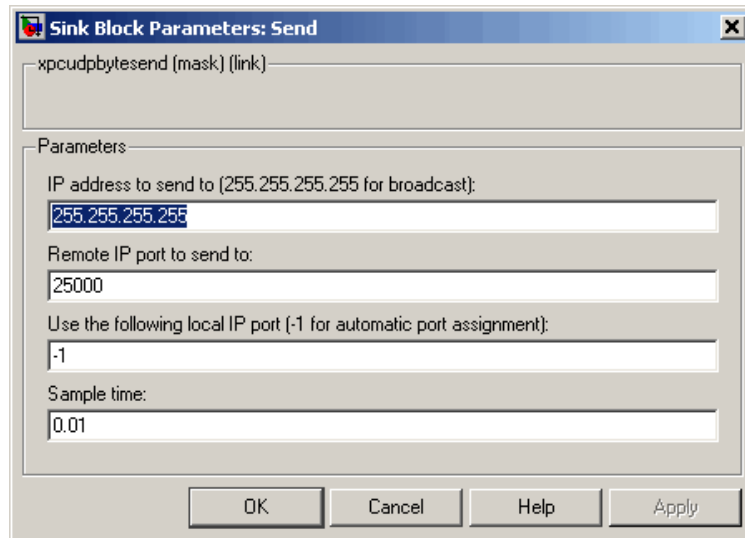
Width of the acceptable packets. You can obtain this when designing the other side (send side) of the communication.

**Sample time**

You can set this parameter to -1 for an inheritable sample time, but it is recommended that this be set to some specific (large) value to eliminate chances of dropped packets. This is especially true when you are using a small base sample time.

# UDP Send

- Purpose** UDP Send block
- Library** xPC Target Library for UDP
- Description** The Send block has only one input port, which receives the uint8 vector that is sent as a UDP packet.



## Block Parameters

- IP address to send to**  
Specify the IP address to send the packet.
- IP port to send to**  
Specify the port to which to send the packet.
- Use the following local IP port**  
Set this parameter to -1 (default) to allow the networking stack to automatically determine the local IP port that is used for sending. Otherwise, specify a particular port to send a packet from that port.



## **Sample time**

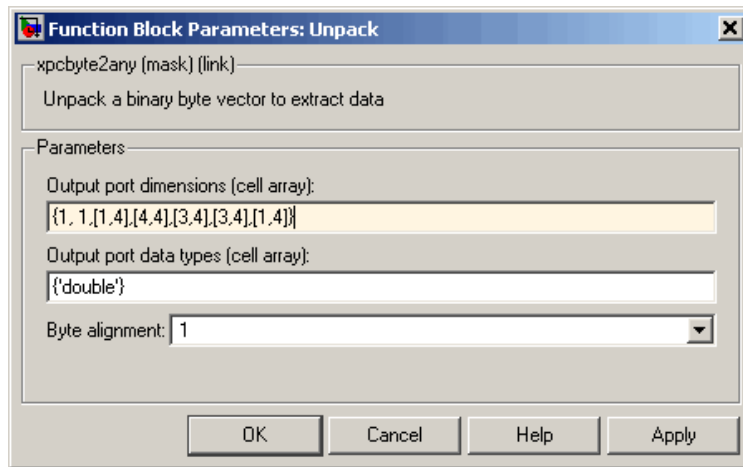
You can set this parameter to -1 for an inheritable sample time, but it is recommended that this be set to some specific (large) value to eliminate chances of dropped packets. This is especially true when you are using a small base sample time.

# UDP Unpack

---

<b>Purpose</b>	UDP Unpack block
<b>Library</b>	xPC Target Library for UDP
<b>Description</b>	This block is the exact analog of the Pack block. It receives a vector of uint8 and outputs various Simulink data types in different sizes. The Pack block is on the sending side and the Unpack block is on the receiving side in different models.
<b>Block Parameters</b>	<p><b>Output port dimensions</b></p> <p>Contains a cell array, with each element the dimension returned by the size function in MATLAB of the corresponding signal. This should normally be the same as the dimensions of the signals feeding into the corresponding Pack block.</p> <p><b>Output port datatypes</b></p> <p>Specify the data types for the different signals as part of the block parameters. The supported data types are double, single, int8, uint8, int16, uint16, int32, uint32, and boolean. This parameter is the same as the <b>Input port data types</b> parameter of the matching UDP Pack block.</p> <p><b>Byte alignment</b></p> <p>The byte alignment field specifies how the data types are aligned. The possible values are: 1, 2, 4, and 8. The byte alignment scheme is simple, and ensures that each element in the list of signals starts on a boundary specified by the alignment relative to the start of the vector.</p>

## Example



As shown in the figure above, the **Output port datatypes** field is the same as the **Input port data types** field of the matching Pack block.

## Note on Byte Alignment

The byte-alignment feature provided in the Pack and Unpack blocks is primarily intended for interfacing a system running xPC Target to another system that is running neither Simulink nor xPC Target. For example, the data on the other end might be in the form of a C struct, which is subject to the byte-alignment convention of the compiler used. We recommend using a byte-alignment value of 1 (tightly packed) whenever possible. Of course, this is easily accomplished when UDP I/O is used to exchange data between two xPC Target systems or between xPC Target and Simulink.

Even when communication is between xPC Target and a system using a C struct, the use of compiler pragmas might help to pack the structure tightly. For example, `#pragma pack(1)` is common to several compilers. The byte-alignment blocks are provided for the case when this is not possible.



# ARINC 429 Support

---

xPC Target interfaces the target PC to an ARINC 429 bus using the ARINC 429 blocks provided by the xPC Target I/O block library. The xPC Target ARINC 429 blocks work with the Condor Engineering CEI-X20 series boards (<http://www.condoreng.com>). This chapter includes the following section:

Condor CEI-520/520A

PCI board series that interfaces target PCs to ARINC 429 data buses.

“Boards and Blocks — Alphabetical List” on page 7-2

Description of the block parameter fields for the xPC Target blocks that support ARINC 429.

## **Boards and Blocks – Alphabetical List**

## Board

Condor CEI-520/520A

## General Description

The commercial and aircraft transport industry uses the ARINC 429 protocol. The ARINC-429 driver library allows xPC Target applications to connect to an ARINC bus network to send and receive 32-bit words. This chapter assumes that you are familiar with the ARINC 429 standard.

xPC Target supports the ARINC 429 protocol with the following Condor boards. These board interface a target PC to an ARINC 429 data bus. These boards support the PCI bus.

- CEI-520
- CEI-520A

xPC Target provides the following driver blocks to support these boards:

- Condor CEI-x20 Initialize
- Condor CEI-x20 Send
- Condor CEI-x20 Receive

Use the following utility blocks to format the data sent to and received from the CEI-x20 Send and Receive blocks:

- Condor Encode ARINC 429 Words for Send
- Condor Decode ARINC 429 Words from Receive

# Condor CEI-520/520A

---

## **Board Characteristics**

Board name	CEI-520, CEI-520A
Manufacturer	Condor
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes



**Purpose** Condor CEI-x20 Initialize block

**Library** xPC Target Library for Arinc-429

**Note** Your model must include a CEI-x20 Initialize block for every physical board in the model. Configure your Send and Receive blocks with the appropriate board ID value from this block to identify the physical board to which they refer. This block supports up to 16 CEI-x20 boards.

**Block Parameters**

**Board ID**

From the list, select from 1 to 16 a unique ID for the CEI board. Use this ID to identify the board in the associated Send and Receive blocks in your model.

**Wrap each send channel to the corresponding receive channel**

Select this check box to enable the hardware loopback feature. If this block is selected, each word sent over the output channel n will be received on the input channel n.

**Timer tick length**

Specify the length of a timer tick in .25 microsecond units. The default value is 4000, which results in a tick length of one millisecond. Time tags (if selected from a Decode ARINC 429 Words from Receive block) are provided in units of timer ticks. A timer tick specifies the units in which the time the time tags is expressed. This concept is provided as a convenience to users.

**Sample time**

Base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

## Condor CEI-x20 Initialize

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** Condor CEI-x20 Send block

**Library** xPC Target Library for Arinc-429

**Note** Use this block to set up one channel of a board to send 32-bit words. The number of channels varies depending on the board you have.

**Block Parameters**

**Board ID**

From the list, select a unique ID from 1 to 16 for the CEI board. Use the ID previously assigned by the associated Initialize block in the model. If you are using a corresponding Receive block for this Send block, select the same board ID as you enter here.

**Channel**

From the list, select a channel ID. This number varies with the particular board you are using. Check your board manufacturer documentation for the number of channels in the board. If you try to select a nonexistent channel, the block returns an error.

**Baud rate**

From the list, select

- 12.5 Kbits/sec
- 100 Kbits/sec

If you are using a corresponding Receive block, be sure to select the same baud rate setting for this channel.

**Parity**

From the list, select

- odd
- none

# Condor CEI-x20 Send

---

---

**Note** If you are using a corresponding Receive block, be sure to select the same parity setting for this channel.

---

## **Sample time**

Base sample time or a multiple of the base sample time.

**Purpose** Condor CEI-x20 Receive block

**Library** xPC Target Library for Arinc-429

**Note** Use this block to set up one channel of a board to receive 32-bit words. The number of channels varies depending on the board you have.

---

**Note** The output port of a Receive block is a signal of type double, but the data on this port is encoded in a nonstandard way. Normally, you should feed this output port into an ARINC Decode block (which converts the data into standard double output). You can also feed it into blocks such as MUX blocks which do not interpret the data. However, before feeding this port into a block such as an xPC Target Scope block which does interpret the data, you must first pass it through an ARINC Decode block.

---

## Block Parameters

### Board ID

From the list, select a unique ID from 1 to 16 for the CEI board. Use the ID previously assigned by the associated Initialize block in the model. If you are using a corresponding Send block, be sure to enter the same board ID.

### Channel

From the list, select a channel number. The number of channels supported varies with the particular board you are using. Check your board manufacturer documentation for the number of channels in the board.

### Max number of words to return

Enter the maximum number of words to extract from the hardware receive buffer. This is the maximum number of words for the selected channel at each sample time.

# Condor CEI-x20 Receive

---

If you select  $n$ , the output port of the block will have a signal width of  $n+1$  and the first signal element will contain the count of words actually extracted from the buffer during the current sample time.

## **Min number of words to return**

Enter the minimum number of words to extract from the hardware receive buffer. This is the minimum number of words for the selected channel at each sample time.

If the hardware receive buffer does not contain this minimum number of words for the selected channel during the current sample time, the block extracts no words from the hardware receive buffer. The word count associated with the output port would then be 0.

## **Baud rate**

From the list, select

- 12.5 Kbits/sec
- 100 Kbits/sec

If you are using a corresponding Send block, be sure to select the same baud rate setting for this channel.

## **Parity**

From the list, select

- odd
- none

If you are using a corresponding Send block, be sure to select the same parity setting for this channel.

## **Sample time**

Base sample time or a multiple of the base sample time.

# Condor Encode ARINC 429 Words for Send

**Purpose** Encode Arinc 429 Words for Send block

**Library** xPC Target Library for Arinc-429

**Note** The output port of an Encode block is a signal of type double. Because the Encode block encodes the data on this port in a nonstandard way, you must send this data to one of the following:

- Condor CEI-520A Send block — The ARINC Send block converts the data into standard double output. This is the block to which you will most likely send data from the ARINC Encode block.
- MUX block — The MUX block does not interpret the data. After the MUX block, you can send it to an ARINC Send block.

## Block Parameters

### Label

Enter a three digit octal number as the label. The label field of each ARINC word sent over the output port will contain this value.

### Data type vector

Enter a vector consisting of values between 0 and 3. These values specify the data type. The length of this vector determines the widths of both the input and output ports. The data type determines how the input double value is converted to a corresponding ARINC value, as follows

Type	Interpretation
0	Raw — Cast the input to an unsigned 32-bit integer and output it as an ARINC word with no further processing.
1	BNR (two's complement binary notation) — Cast the input as a signed integer, clamp it to be in the range representable by a signed 19-bit binary integer, and pack it into an ARINC word with the appropriate Sign/Status Matrix (SSM), Source/Destination Identifier (SDI), and Label data.

# Condor Encode ARINC 429 Words for Send

Type	Interpretation
2	BCD (binary coded decimal) — Cast the input as a signed integer, clamp it to be in the range representable by an ARINC five character BCD value, and pack it into an ARINC word with the appropriate SSM, SDI, and Label data.
3	Discretes — Cast the input as an unsigned 32-bit integer and pack the low order 19 bits of the result into an ARINC word with the appropriate SSM, SDI, and Label data.

## Resolution vector

Enter a vector or scalar value as the resolution vector. This must be a vector of the same length as the **data type vector**. Otherwise, the scalar value is applied to the length of the **data type vector**. The block works with the data types as follows

Type	Effect
Raw	The block ignores any resolution value. However, you must still include an associated value in the resolution vector.
BNR	<p>The resolution value specifies, in the same units as the input signal, the value of the least significant bit of the binary data field. For example, if the associated resolution value is 10 and the input signal contains the value 1000, the output ARINC word will contain the binary number 100 in its data field.</p> <p>A 19-bit signed ARINC binary can represent a range from -262,144 to 262,143. If the combination of input signal and resolution produces a value outside this range, the block clamps it to be within the range.</p>



# Condor Encode ARINC 429 Words for Send

Type	Effect
BCD	<p>The resolution value specifies, in the same units as the input signal, the value of the least significant digit of the BCD data field. For example, if the associated resolution is 10 and the input signal contains the value 1000, the output ARINC word will contain the number 100 in its data field, encoded in BCD.</p> <p>The range representable as an ARINC BCD value is -79,999 to 79,999. If the combination of input signal and resolution produces a value outside this range, the block clamps it to be within the range.</p>
Discretetes	The block ignores any resolution value. However, you must still include an associated value in the resolution vector.

## SDI vector

Enter a vector or scalar value as the SDI vector. This must be a vector of the same length as the **data type vector**. Otherwise, the scalar value is applied to the length of the **data type vector**.

This block interprets the **SDI vector** values as follows:

Type	Effect
Raw	The block ignores any resolution value. However, you must still include an associated value in the resolution vector.
BNR, BCD, Discretetes	If the SDI element is in the range 0 to 3, the block sets the SDI field of the corresponding output word to that value. If an SDI element has a value of -1, the block performs no SDI processing on the corresponding output word.

# Condor Encode ARINC 429 Words for Send

---

## **SSM vector**

Enter a vector or scalar value as the SDI vector. This must be a vector of the same length as the **data type vector**. Otherwise, the scalar value is applied to the length of the **data type vector**.

If the SSM element is in the range 0 to 3, the block sets the SSM field of the corresponding output word to that value. If an SSM element has a value of -1, the block performs no SSM processing on the corresponding output word. Note that the meaning of a given SSM value differs depending on the data type of the ARINC word.

# Condor Decode ARINC 429 Words from Receive

---

**Purpose** Decode ARINC 429 Words from Receive block

**Library** xPC Target Library for Arinc-429

**Note** The input port of a Decode block is a signal of type double. Because the Decode block interprets the data on this port in a nonstandard way, you can send the data to this port from one of the following:

- Condor CEI-520A Receive block — The Condor CEI-520A Receive block outputs its data into standard double output. This is the block from which you will most likely send data to the ARINC Decode block.
- ARINC Encode block — The output port of an Encode block is a signal of type double.

The output port of a Decode block is in standard double format.

The input to this block should be a CEI-x20 Receive block. The Decode block input port width will automatically adapt to that of the source block. The connected width of the input port of the block adjusts to the block it is connected to.

## Block Parameters

### Label

Enter a three digit octal number. If the label of an input word does not match this label, the block completely ignores the word and does not apply the **Sync mask** and **Sync value** parameters.

### Data type vector

Enter a vector consisting of values between 0 and 3. These values specify the data type. The length of this vector determines how many ARINC words the block will attempt to decode and output each sample time.

The elements of the vector determine how the input double value is converted to a corresponding double output value, as follows

# Condor Decode ARINC 429 Words from Receive

Type	Interpretation
0	Raw — Convert the entire (unsigned) 32 bit input word to double.
1	BNR — For each word, convert bits 10-28 from signed binary format to double.
2	BCD — For each word, convert bits 10-28 from BCD format to double, using the sign data in the SSM.
3	Discretes — For each word, extract bits 10-28 and return them as a double.

The elements of the data type vector determine how the input ARINC value is converted to a corresponding double output. The following describes how this block performs the conversion. For the purposes of this description,  $n$  denotes the length of the data type vector.

- The output width is one of the following, depending on your time tag selection
  - $2n + 1$  — This is the output width if you select the **Provide time tags** check box. The width consists of a count element, followed by  $n$  data elements, followed by  $n$  time tag elements.
  - $n + 1$  — This is the output width if you do not select the **Provide time tags** check box. The width consists of a count element followed by  $n$  data elements.

The count element indicates how many valid messages the block has decoded during the current sample time. The count element has a nonzero value if at least one message on the data element is currently valid, and zero otherwise. If the count element has a value greater than one, the block asserts only the most recent valid message on the output port.

# Condor Decode ARINC 429 Words from Receive

- The Decode block buffers its input internally. It updates its output port only during sample times when it has successfully assembled at least one complete message of length n.

Note that more than one complete message might be assembled during one sample time. The Decode block successively overwrites these messages such that only the most recent message is on the output port.

## Resolution vector

Enter a vector or scalar value as the resolution vector. This must be a vector of the same length as the **data type vector**. Otherwise, the scalar value is applied to the length of the **data type vector**. The block works with the data types as follows

Type	Effect
Raw	The block ignores any resolution value. However, you must still include an associated value in the resolution vector.
BNR, BCD	The resolution value specifies, in output port units, the value of the least significant bit of the data field. For example, if the resolution is 10 and the input data contains the value 100, the output signal will contain the number 1000.
Discretes	The block ignores any resolution value. However, you must still include an associated value in the resolution vector.

## Sync mask

Enter a value, in hexadecimal, to specify which bits (if any) of the input words are the sync bits. (A sync bit lets you specify, using other parameters, when a message should begin.) The Decode block will examine these bits to look for the start of the next message. A message might be a string of one or more words. For example, a sync mask value of 0x300 equals 110000000 in binary. This value selects the SDI bits (bits 9 and 10) as the sync

# Condor Decode ARINC 429 Words from Receive

---

bits. This functionality works in conjunction with the **Sync value** parameter.

If the sync mask is 0x0, no sync logic is used. In this case, the next word always begins a new message.

## **Sync value(s)**

This parameter specifies the sync logic for the block. Enter one hex value to specify oneSync, two hex values separated by a space for twoSync logic. For example, the sync value

0x100

selects oneSync logic. The sync value

0x100 0x200

selects twoSync logic. You can enter an 32-bit value.

The sync value takes into account the value of the sync mask, as follows:

- Assume the following:

Sync mask = 0x300

Sync value = 0x100

When looking for the beginning of a new message, the block ANDs each input word with the sync mask 0x300 and compares the result with 0x100 and 0x300. When it finds a match, the block stops this search and begins a new search, looking for the next message. The block decodes the next n words starting at this point.

- Assume the following:

Sync mask = 0x300

Sync value = 0x100 0x200

When looking for the beginning of a new message, the block ANDs each input word with 0x300 and compares the result to 0x100 and

# Condor Decode ARINC 429 Words from Receive

---

0x300. When the block finds a match, and when the next input word, when ANDed with 0x300, equals 0x200 and 0x300, this second word begins a new message.

Once the block locates the beginning of a message, it uses the next  $n$  input words with the appropriate label to assemble the next output message. The block does not use sync logic until it is time to begin the assembly of a new message.

## **Provide time tags**

Select this check box to enable an output port of width  $2n + 1$ , with time tag data in the last  $n$  elements.





# MIL-STD-1553 Support

---

xPC Target interfaces the target PC to an MIL-STD-1553 bus using the MIL-STD-1553 blocks provided by the xPC Target I/O block library. The xPC Target MIL-STD-1553 blocks work with the Condor Engineering PCI-1553 and QPCI-1553 series boards (<http://www.condoreng.com>). This chapter includes the following sections:

“Introduction” on page 8-2

Introduction to the xPC Target I/O block library that supports the MIL-STD-1553 bus with the MIL-STD-1553 sublibrary.

Condor PCI/QPCI-1553

PCI and QPCI board series that interfaces target PCs to MIL-STD-1553 data buses.

“Boards and Blocks — Alphabetical List” on page 8-13

Description of block parameters for MIL-STD-1553 driver blocks.

## Introduction

The xPC Target I/O block library supports the MIL-STD-1553 bus with the MIL-STD-1553 sublibrary. This sublibrary supports the Condor boards, QPCI-1553 and PCI-1553. The sublibrary consists of the following groupings

- 1553 Utilities — Use these general utility blocks to
  - Set up Bus Monitor and Bus Controller messages
  - Create Bus Controller message lists
  - Encode and decode bus controller messages and status
- PCI/QPCI-1553 — Use these blocks to communicate with the PCI/QPCI-1553 boards. These blocks enable you to
  - Initialize a board for Remote Terminal, Bus Controller, and/or Bus Monitor operation.
  - Configure a board for Remote Terminal operation, including Remote Terminal initialization and sending/receiving messages
  - Configure a board to send Bus Controller messages

The following topics provide guidelines on how you can use the xPC Target MIL-STD-1553 blocks for the following operations:

- “Remote Terminal Operation” on page 8-5
- “Bus Controller Operation” on page 8-7
- “Remote Terminal and Bus Controller Operation” on page 8-9
- “Bus Monitor Operation” on page 8-10

The models for these examples are located in the `xpcdemos` directory.

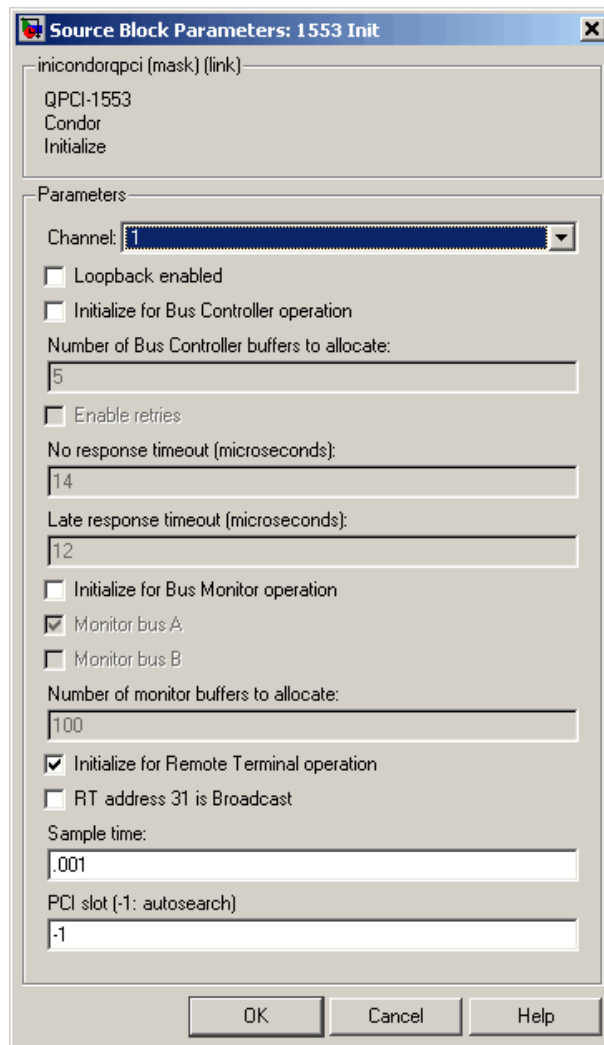
### Before You Start

Regardless of the operation, always initialize your board with the PCI/QPCI Initialize block. (Be sure to use the appropriate board block grouping.) The examples in this topic use the QPCI blocks.

The QPCI Initialize block allows you to specify the board operation, Remote Terminal, Bus Controller, and/or Bus Monitor operation. The dynamic dialog for this block changes depending on the operation you select.

- Select the **Initialize for Bus Controller operation** check box for the Bus Controller
- Select the **Initialize for Bus Monitor operation** check box for the Bus Monitor
- Select the **Initialize for Remote Terminal operation** check box for the Remote Terminal

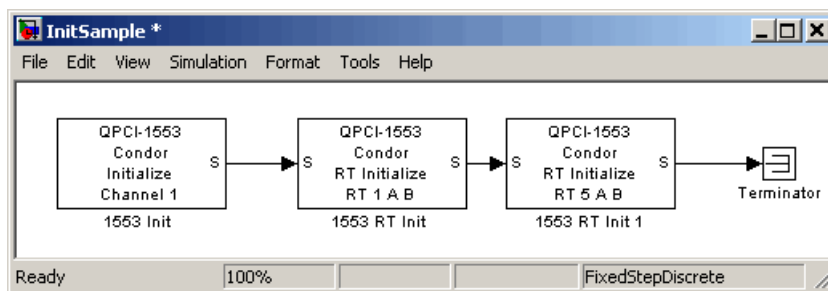
By default, these check boxes are selected. If you deselect an operation check box, the block grays out the associated parameters. For example,



The following is a sample model of how to use the QPCI-1553 Initialize block to initialize channel 1 of a board for Remote Terminal operation. This model

- Configures two QPCI-1553 Remote Terminal Initialize blocks, one for Remote Terminal 1 and Remote Terminal 5 on that channel.
- Initializes each Remote Terminal with legal subaddresses and legal message lengths for each subaddress.

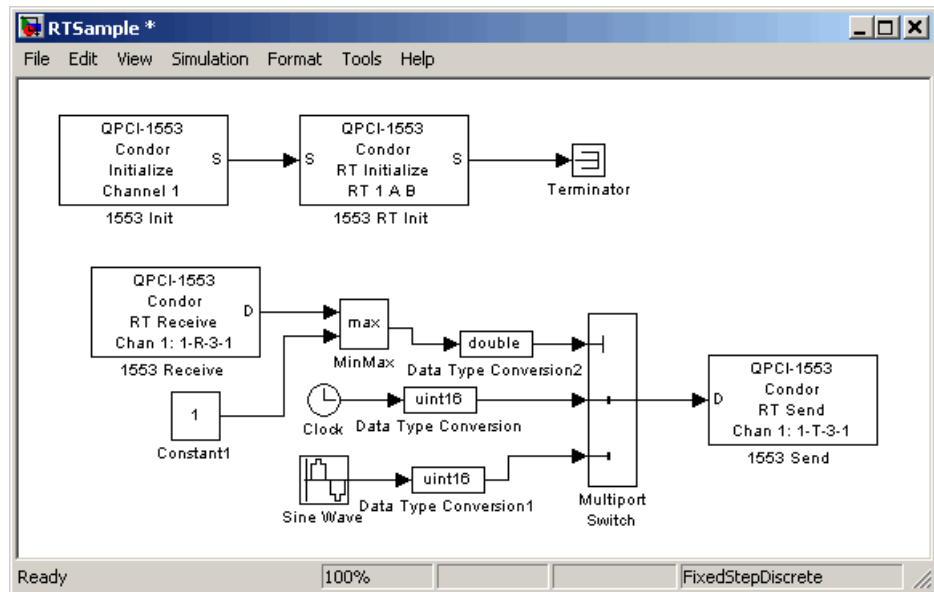
You can configure each subaddress for transmit, receive, or both. Configure all the subaddresses you plan to use.



## Remote Terminal Operation

The example in this topic uses QPCI-1553 blocks to illustrate how you can configure a Remote Terminal from an xPC Target model. For standard initialization use the QPCI Initialize block. Use the QPCI RT Initialize block to set up the board for Remote Terminal operation.

Run this example on a target PC that has a QPCI-1553 board. Replace the hardware specific blocks with the correct blocks if your hardware configuration is different.



The QPCI-1553 Initialize block configures the board for Remote Terminal operation on channel 1 of the board. The QPCI-1553 RT Initialize block configures the Remote Terminal 1 to monitor buses A and B for incoming messages and configure the active transmit and receive subaddresses. The QPCI-1553 Receive and Send blocks use subaddress 3 for both transmit and receive. Note the message parameters notation 1-R-3-1 on the QPCI-1553 Receive and Send blocks. This notation has the format

remote terminal-R/T-subaddress-number of words

Remote terminal — Indicates the particular Remote Terminal

R/T — Indicates that the command is for receive (R) or transmit (T)

Subaddress — Indicates the subaddress for the message

Number of words — Indicates the number of 16 bit integers to receive as the data part of a message.

This is shorthand that indicates that Remote Terminal 1 is to receive messages on subaddress 3 with a length of 1 word. The incoming message consists of one 16-bit integer with a value of 1 or 2. The Multiport Switch block uses this input to select either clock or sine wave data.

A Condor board has a processor that handles the actual transmission and reception of messages. The QPCI-1553 Receive block reads the board receive message buffer for the specified Remote Terminal and subaddress. The QPCI-1553 Send block writes data to the board transmit buffer.

---

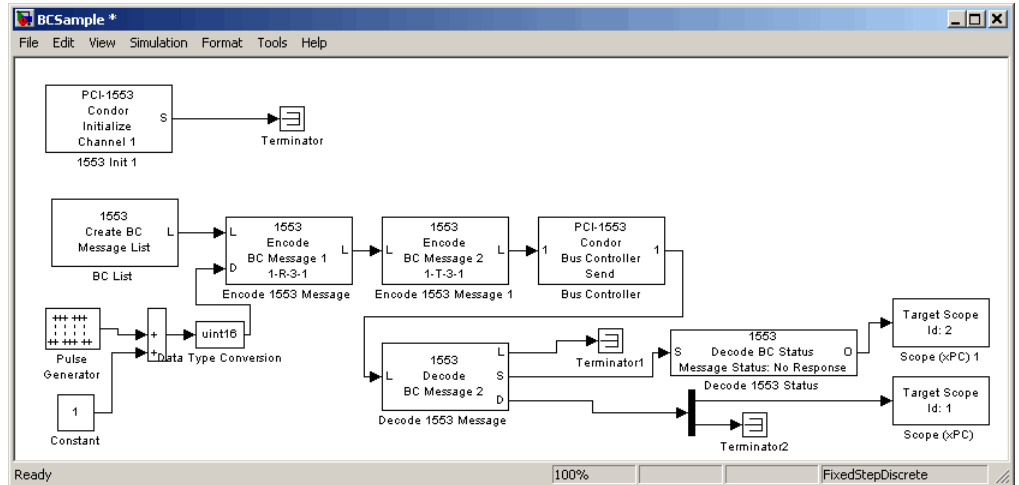
**Note** When the board receives a transmit or receive message, it must respond quickly because the default timeout is 14 microseconds. This timeout is much quicker than the ability of xPC Target to reliably execute a model. This situation might cause the board to transmit data from the last time the model executed. Design your model accordingly.

---

## Bus Controller Operation

The example in this topic uses PCI-1553 blocks. It describes how you can use a Bus Controller in a model. This model is a simple example of how to set up a short sequence of two messages, send them, and collect the response data.

Run this example on a target PC that has a PCI-1553 board. Replace the hardware specific blocks with the correct blocks if your hardware configuration is different.



The PCI-1553 Initialize block configures a Bus Controller on channel 1 of the board. This block also tells the board to reserve five message buffers. This number must be at least as large as the longest list of messages that you will send.

The Create BC Message List block allocates an empty list of five message buffers in xPC Target memory. Each time this block executes, it sets the entire list of message buffers to no-op messages. The Encode BC Message block fills each message for any timestep during which the message should be sent. You can also use the Encode BC Message block in an enabled subsystem to only send that message when desired.

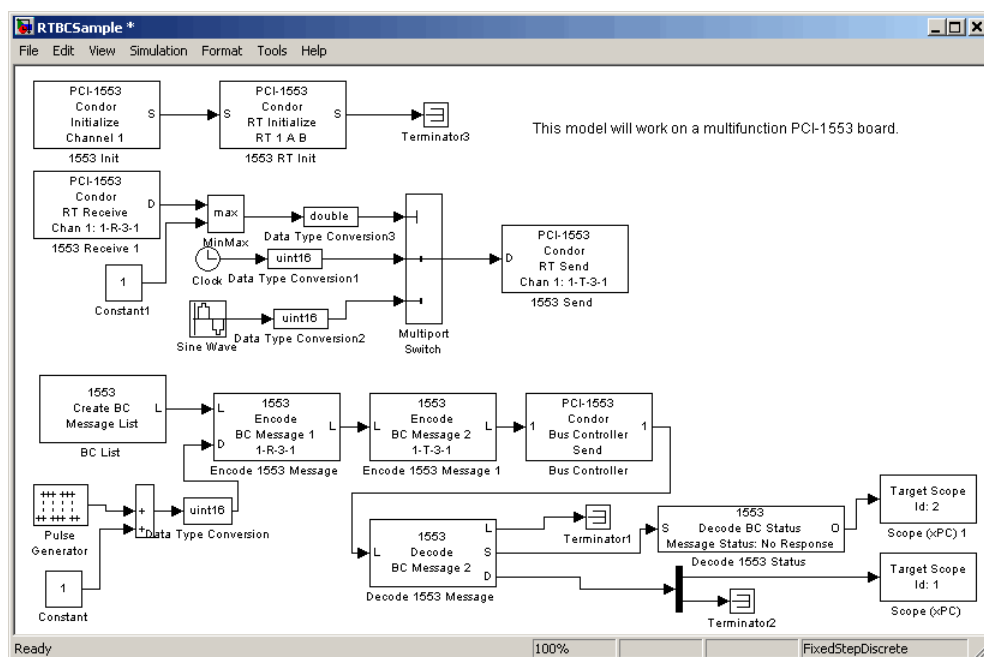
The L signal is the message list. The model passes this signal through all the Encode BC Message blocks, then connects to the Bus Controller Send block. This ensures that the model executes all the Encode BC Message blocks execute before the Bus Controller Send block. The L signal is a custom data type consisting of a pointer, message length, and special marker.



## Remote Terminal and Bus Controller Operation

The example in this topic uses PCI blocks. It describes how you can use a Remote Terminal and Bus Controller in a model. This model combines the three models, RTInit.mdl, RTSample.mdl and BCSample.mdl.

RTBCSample runs on a target PC with a multifunction board. It shows how to configure and use a single board as a Bus Controller and Remote Terminal. You can also run this model in one target PC with two different Condor 1553 boards.



In this example, the two Encode BC Message blocks each create a message, for a total of two, in the list. The message created in the first Encode BC Message block is a receive message. Note, the direction of a message is always from the point of view of the Remote Terminal. The **Direction** parameter of the block has a value of R (BC->RT). The 1553 Encode BC Message block creates message 1, one 16-bit word, to be sent to Remote Terminal 1, subaddress 3. This word has a value of either 1 or 2. This is because the Pulse Generator outputs either 0 or 1, to which a constant value 1 is added.

Message 2 of Bus Controller message list is a command for Remote Terminal 1, subaddress 3, to send one word. From the RTSample model, that data word comes back containing either the clock or the sine wave data.

The PCI-1553 Bus Controller Send block takes a fully formed list of message buffers and sends it. It waits for the messages to be sent and the response to be received. The PCI-1553 Bus Controller Send block has a programmable maximum wait time parameter (**Maximum wait time**). In this model, the maximum wait time is 1000 microseconds.

The response message list from the PCI-1553 Bus Controller Send block has the same length as the list that was sent. The responses are found in the same position in the list as the corresponding command. In this case, the command to the Remote Terminal to send data is message number 2. The data that is returned is also message number 2 in the response list. In this example, the Decode BC Message block check message 2 only. Decode BC Message blocks can be in any order on the list. To avoid confusion, put them in numerical order. This example does not check message 1. You can add another Decode BC Message block to check message 1. In this case, only the status is useful because the data is exactly what was sent.

The Decode BC Message block has the following outputs.

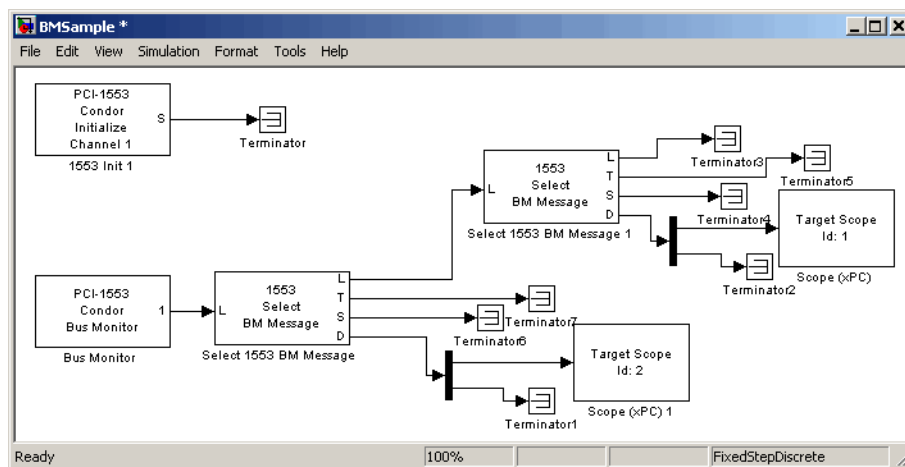
- L — Message list passed to other Decode BC Message blocks. If there are no other Decode BC Message blocks, connect the signal to a terminator or ground.
- S — Status information. The Decode BC Status block extracts individual status bits from this status. To get multiple status bits, use multiple Decode BC Status blocks and feed, in parallel, the same signal to the S ports.
- D — A vector of 32 short integers with the data from that message. Output from the Decode BC Message block always has the maximum message width of 32 uint16 values. The actual message determines how many of these are significant. In this example, only the first value is significant. The block sends the last 31 values to a terminator.

## **Bus Monitor Operation**

This topic uses PCI-1553 blocks. It describes how you can use a Bus Monitor in a model. The Bus Monitor operation outputs a list containing all the messages

seen on the specified bus since the last time it was called. The BMSample model show how to set up a simple bus monitor that looks for the messages in the preceding examples (RTSample.mdl, BCSample.mdl, and RTBCSample.mdl).

Run this example on a target PC that has a PCI-1553 board. Replace the hardware specific blocks with the correct blocks if your hardware configuration is different.



The PCI-1553 Initialize block configures a Bus Monitor on channel 1 of the board. This block also tells the board to monitor bus A. This PCI-1553 Bus Monitor block specifies a maximum of 10 messages to receive in a list.

The Select BM Message block picks a message with specified properties out of this list. The Message selection mode parameter of the block provides a finite list of message properties. In this model, the selected property is BC->RT or RT->BC.

The Select BM Message block has the following outputs:

L — Message list passed to other Select BM Message blocks. If there no other Select BM Message blocks, connect the signal to a terminator or ground.

T — Time that the board believes that the message was received on the bus. This is the time since the board was started in microseconds, presented as

a double. The clock might run at a slightly different rate than the model execution timer. This implies that this time is likely different from the xPC Target execution time.

S — Status information. This contains seven uint32 entries with status and command information.

D — A vector of 32 uint16 entries with the data from that message. This always outputs all 32 entries, even if only some of the entries are defined.

## **Boards and Blocks – Alphabetical List**

# Condor PCI/QPCI-1553

---

## Board

Condor PCI/QPCI-1553

## General Description

The commercial and aircraft transport industry use the MIL-STD-1553 protocol. The MIL-STD-1553 driver library allows xPC Target applications to connect to an MIL-STD-1553 bus network to send and receive messages of up to 32 16-bit words. This chapter assumes that you are familiar with the MIL-STD-1553 standard.

xPC Target supports the MIL-STD-1553 protocol with the following Condor boards. These board interface a target PC to an MIL-STD-1553 data bus. These boards support the PCI and high-density PCI (QPCI) bus.

- Condor PCI-1553 — This is available with 1 or 2 channels as either a single function or multi-function configuration. The multi-function version supports simultaneous use of the Bus Controller, Bus Monitor, and Remote Terminal functions. The single function version emits an error if more than one function is initialized. PCI-1553 blocks support up to two channels for this board.
- Condor QPCI-1553 — This board is available with 1, 2, or 4 channels as either a single function or multi-function configuration. The multi-function version supports simultaneous use of the Bus Controller, Bus Monitor, and Remote Terminal functions. The single function version emits an error if more than one function is initialized. This board also supports a loopback mode for testing. QPCI-1553 blocks support up to four channels for this board.

xPC Target provides the following driver blocks to support these boards:

- Condor PCI-1553 Bus Controller Send
- Condor PCI-1553 Bus Monitor
- Condor PCI-1553 Initialize
- Condor PCI-1553 RT Initialize
- Condor PCI-1553 RT Receive

- Condor PCI-1553 RT Send
- Condor QPCI-1553 Bus Controller Send
- Condor QPCI-1553 Bus Monitor
- Condor QPCI-1553 Initialize
- Condor QPCI-1553 RT Initialize
- Condor QPCI-1553 RT Receive
- Condor QPCI-1553 RT Send

Use the following utility blocks to format the Bus Controller messages:

- Condor 1553 Create BC Message List
- Condor 1553 Decode BC Message
- Condor 1553 Decode BC Status
- Condor 1553 Encode BC Message
- Condor 1553 Select BM Message

## Board

## Characteristics

Board name	PCI/QPCI-1553
Manufacturer	Condor
Bus type	PCI
Multiple board support	Yes

# Condor 1553 Create BC Message List

---

<b>Purpose</b>	Condor Create Bus Controller (BC) Message List block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Description</b>	This block allocates space for a BC message list. It sets all messages in the list to no-op. This implies that the execution of a message on a given time step requires that a 1553 Encode BC Message block execute on that time step. The 1553 Create BC Message List block performs a clearing operation that allows you to control when a message is sent by placing 1553 Encode BC Message blocks in enabled subsystems.
<b>Block Parameters</b>	<b>Number of message buffers</b> Enter the number of message buffers for the list. This block allocates an empty list for these buffers in xPC Target memory. <b>Sample time</b> Enter the base sample time or a multiple of the base sample time.
<b>See Also</b>	Condor 1553 Encode BC Message “Bus Controller Operation” on page 8-7



# Condor 1553 Decode BC Message

<b>Purpose</b>	Condor 1553 Decode Bus Controller (BC) Message block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Description</b>	<p>Each instance of this block extracts information out of one message in the receive list. This block has three outputs.</p> <p>L — Specifies the input and output message list passed to other Decode BC Message blocks. If there are no other Decode BC Message blocks, connect the signal to a terminator or ground.</p> <p>S — The S output is a vector of six elements from the received message, all bit packed with information:</p> <p style="padding-left: 40px;">control, command1, status1, command2, status2, overall status</p> <p>These elements derive from the Condor API_BC_MBUF structure, which is contained in the xPC Target file <code>matlab\toolbox\rtw\targets\xpc\target\build\xpcblocks\include\busapi.h</code>. See the Condor 1553 software user documentation for details on this structure.</p> <p>Use the Decode BC Status block to extract individual status bits from this status. To get multiple status bits, use multiple Decode BC Status blocks and feed, in parallel, the same signal to the S ports.</p> <p>D — The D output is a vector of 32 short integers with the data from that message. For a message sent to an RT (R direction), these values are the ones sent. For a message from an RT (T direction), this contains the values from the RT. You do not need to decode the received message from an R message.</p>
<b>Block Parameters</b>	<p><b>Message number</b> Enter the number of the message to decode.</p> <p><b>Sample time</b> Enter the base sample time or a multiple of the base sample time.</p>

# Condor 1553 Decode BC Status

---

<b>Purpose</b>	Condor 1553 Decode Bus Controller (BC) Status block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Description</b>	This block extracts individual status bits from the status output of the Decode BC Message block.
<b>Block Parameters</b>	<p><b>Status to read</b> From the list, choose a status to read. The Decode BC Status block extracts individual status bits from the status (S) output of the Decode BC Message block.</p> <p><b>Sample time</b> Enter the base sample time or a multiple of the base sample time.</p>

# Condor 1553 Encode BC Message

<b>Purpose</b>	Condor 1553 Encode Bus Controller (BC) Message block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Description</b>	<p>This block fills each message for any timestep during which the message should be sent.</p> <p>The Create BC Message List block allocates an empty list of message buffers in xPC Target memory. Use the Encode BC Message block to fill these messages.</p>
<b>Block Parameters</b>	<p><b>Message number</b> Enter the number of the message to encode. Each BC message needs to use a unique message number. This is the place in the list to fill with this message.</p> <p><b>Remote Terminal address 1 (Receive if RT-RT)</b> From the list, choose a Remote Terminal address 1.</p> <p><b>Sub address 1 (Receive if RT-RT)</b> From the list, choose a Remote Terminal subaddress 1. Setting this parameter to Send (0) mode command or Send (31) mode command disables the <b>Message word count</b> parameter.</p> <p><b>Remote Terminal address 2 (Transmit if RT-RT)</b> From the list, choose a Remote Terminal address 2. This is not used if the <b>Direction</b> is R (BC-&gt;RT) or T (RT-&gt;BC).</p> <p><b>Sub address 2 (Transmit if RT-RT)</b> From the list, choose a Remote Terminal subaddress 2. This is not used if the <b>Direction</b> is R (BC-&gt;RT) or T (RT-&gt;BC).</p> <p><b>Mode command</b> If the <b>Sub address 1 is 0 or 31</b>, from the list, choose a mode command.</p> <p><b>Message word count</b> Specify the number of uint16 words to send or receive.</p>

# Condor 1553 Encode BC Message

---

## **Direction**

From the list, choose the encode direction. Choose from

- R (BC->RT)
- T (RT->BC)
- RT->RT

## **Inter message gap to next message (usec)**

Specify the minimum amount of time, in microseconds, between messages.

## **Output bus**

From the list, choose output bus A or B that this message will be sent on.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **See Also**

Condor 1553 Create BC Message List, Condor 1553 Decode BC Message

# Condor 1553 Select BM Message

---

<b>Purpose</b>	Condor Select 1553 Bus Monitor (BC) Message block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Description</b>	This block picks a message with specified properties out of a Bus Monitor list from the PCI-1553 Bus Monitor block.

## Block Input and Outputs

This block has the following inputs and outputs:

- L — The passed through message list
- T — Time in microseconds at which this message was received
- S — Status (seven component vector with contents)

- 1 message number
- 2 cmd1
- 3 resp1
- 4 status 1
- 5 cmd 2
- 6 resp 2
- 7 status 2

Note that cmd1 and cmd2 contain address information in bit fields in the 16-bit integer.

cmdN=<RRRRR>T<SSSS><CCCC>

Where RRRRR is the 5-bit field with the remote terminal address. T is 1 if a transmit message, 0 if receive. S is the subaddress. C is the count.

Note that 5, 6, and 7 are only non-zero for the RT->RT messages.

D — 32 uint16 vector data. cmd 1 has the length of real data in the low order 5 bits.

# Condor 1553 Select BM Message

---

The block clears all status and data words to 0 if a matching message is not found in the list.

## Block Parameters

### Message selection mode

From the list, choose a message selection (filtering) mode

- message number — By message number
- BC->RT or RT->BC — By Remote Terminal and subaddress
- RT->RT — By both Remote Terminal and subaddresses

### Message number

Specify an index into a Bus Monitor list. Use this parameter if you set **Message selection mode** to message number.

### Remote Terminal 1

Specify the Remote Terminal from which to select the message. Use this parameter if you set **Message selection mode** to one of the following:

- BC->RT or RT->BC
- RT->RT (receive side)

### Sub address 1

Specify the subaddress from which to select the message. Use this parameter if you set **Message selection mode** to one of the following:

- BC->RT or RT->BC
- RT->RT (receive side)

### Remote Terminal 2

Specify the Remote Terminal from which to select the message. Use this parameter if you set **Message selection mode** to RT->RT (send side).

# Condor 1553 Select BM Message

---

## **Sub address 2**

Specify the subaddress from which to select the message. Use this parameter if you set **Message selection mode** to RT->RT (send side).

## **Direction**

From the list, choose the encode direction. Use this parameter if you set **Message selection mode** to BC->RT or RT->BC. Choose from

- R (BC->RT) — Receive, from Bus Controller to Remote Terminal
- T (RT->BC) — Transmit, from Remote Terminal to Bus Controller

## **Sample time**

Enter the base sample time or a multiple of the base sample time. Enter -1 to inherit the sample time.

## **See Also**

Condor PCI-1553 Bus Monitor

# Condor PCI-1553 Bus Controller Send

---

**Purpose** Condor PCI-1553 Bus Controller (BC) Send block

**Library** xPC Target Library for MIL-STD-1553

**Description** This block sends a list of messages to the specified channel on the board. Before you construct the list of messages:

- 1 Allocate space for the list with the Condor 1553 Create BC Message List block.
- 2 Fill in information about each message with Condor 1553 Encode BC Message blocks.

Each instance of an Encode BC Message block fills in information for a single message on the list. To ensure that all Encode BC blocks execute before the Bus Controller Send block, daisy chain the Encode BC Message block L signals.

The Bus Controller Send block allocates a list for the reception of messages. This block can

- Read the set of messages and statuses from the last time a message was sent before sending the new list.
- Send the new list and wait for a response.

In other words, for time step N, the output of the Send block is either the response to time step N-1 or the response to time step N, chosen from the **Response mode** parameter.

## Block Parameters

### Channel

From the list, choose 1 or 2. This is the channel for this command stream.



# Condor PCI-1553 Bus Controller Send

---

## Response mode

Choose from

- **Read response to previous message then send new message** — Read the command buffers from the board before sending the new ones.
- **Send new message then wait and read response** — Transmit new message, then wait for response before continuing.

See the **Maximum wait** time parameter to set the amount of time the block should wait for a response.

## Maximum wait time (microseconds)

Enter the maximum time that this block waits for the message stream to be sent by the board. Use this parameter if the **Response mode** parameter is set to **Send new message then wait and read response**.

A reasonable maximum wait time is 1000 microseconds (1 millisecond). Note, this value should not exceed the sample time except for failsafe instances. Exceeding the sample time will trigger an execution overload.

## Sample time

Enter the base sample time or a multiple of the base sample time.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Condor PCI-1553 Bus Controller Send

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose**

Condor PCI-1553 Bus Monitor block

**Library**

xPC Target Library for MIL-STD-1553

**Block  
Parameters****Channel**

From the list, choose 1 or 2.

**Maximum number of messages to receive**

Enter the maximum number of messages this block should read from the board.

**Sample time**

Enter the base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Condor PCI-1553 Initialize

---

**Purpose** Condor PCI-1553 Initialize block

**Library** xPC Target Library for MIL-STD-1553

**Block Parameters** **Channel**

From the list, select from 1 or 2. This is the channel to initialize on the board.

## **Initialize for Bus Controller operation**

Select this check box to perform Bus Controller initialization for this channel. Selecting this check box enables the next four parameters.

- **Number of Bus Controller buffers to allocate** — Enter the number of buffers to allocate in onboard memory on the board. This number must be greater than or equal to the longest string of command buffers that the board will be given to process.
- **Enable retries** — Select this check box to enable automatic retries when a message times out or has an error.
- **No response timeout (microseconds)** — Enter the number of microseconds the board is to wait for a response from the Remote Terminal. If the response from the Remote Terminal takes longer than this timeout value, the board sets the NORESPONSE error condition in the status returned from the command.
- **Late response timeout (microseconds)** — Enter the number of microseconds the board is to wait for a response from the Remote Terminal. If the response from the Remote Terminal takes longer than this timeout value, the board sets the LATERESPONSE error condition in the status returned from the command.

## **Initialize for Bus Monitor operation**

Select this check box to initialize this channel for Bus Monitor operation. You must select this check box to initialize the channel for Bus Monitor operation. Selecting this check box enables the parameters:

- **Monitor bus A** — Select this check box to monitor bus A. You can monitor either bus A, bus B, or both.
- **Monitor bus B** — Select this check box to monitor bus B. You can monitor either bus B, bus A, or both.
- **Number of monitor buffers to allocate** — Enter the number of at least as many buffers as you expect to see messages between calls to the Bus Monitor block. If more messages arrive than there are buffers, some messages will be lost and not monitored.

### **Initialize for Remote Terminal operation**

Select this check box to prepare the board to operate as a Remote Terminal. If you select this check box, add the RT Initialize block to the model to initialize the Remote Terminal. Connect the RT Initialize block for the Remote Terminal to the S output of this block.

Selecting this check box enables the parameter:

### **RT address 31 is Broadcast**

Select this check box to enable the board to see messages to Remote Terminal 31 as broadcast messages.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Condor PCI-1553 RT Initialize

---

**Purpose** Condor PCI-1553 RT Initialize block

**Library** xPC Target Library for MIL-STD-1553

**Description** Add an RT Initialize block for each Remote Terminal address for which you want this board to accept commands. Because each of these blocks must execute after the global board initialization, connect the S input to either

- Main Condor PCI-1553 Initialize block for this board
- Another Condor PCI-1553 RT Initialize block that is connected to the main Condor PCI-1553 Initialize block.

This connection carries the channel number and PCI slot information.

## **Block Parameters**

### **Remote Terminal**

From the list, choose a Remote Terminal from 1 to 31. Choose the terminal number that this initialization block is configuring.

### **Connect to bus A**

Select this check box to have this Remote Terminal listen on bus A. Choose both A and B if a message can come in on either bus.

### **Connect to bus B**

Select this check box to have this Remote Terminal listen on bus B. Choose both A and B if a message can come in on either bus.

### **Initial status**

Enter the initial value of the status before any commands are executed. The board maintains a status word that is sent to the Bus Controller after command execution.

### **Initial BIT word**

Enter the initial value of Built In Test (BIT) word. The Remote Terminal sends this word in response to the Transmit BIT Word mode code.

## Inhibit terminal flag

Select this check box to inhibit the Remote Terminal flag. This is the initial value of the Inhibit Terminal Flag bit. This can be changed from the Bus Controller with the Inhibit Terminal and Override Inhibit Terminal mode codes. The terminal bit is one of the hardware maintained bits in the status word.

## Transmit sub addresses

Enter a vector of subaddresses to which this Remote Terminal should respond. These are the subaddresses that can transmit data when requested. Each element of the vector must correspond to the corresponding element of the **Legal transmit message lengths** vector. To enable the subaddress for the Remote Terminal, include the subaddress in the vector.

A Remote Terminal only accepts a message if it has been directed to accept messages of the same length. During initialization, the board must receive a list of valid message lengths for each Remote Terminal number.

## Legal transmit message lengths

Enter a vector of transmit message lengths. This is a vector of bit masks where each bit corresponds to a single message length. Ensure the following:

- Each element of the vector must correspond to the corresponding element of the **Transmit sub addresses** vector.
- To enable a transmit message, each element of the vector must correspond to the corresponding element of the **Legal receive message lengths**. Failure to do this prevents the subaddress on this Remote Terminal from responding.

Because a message can have from 1 to 32 values in it, the block uses a single 32-bit integer to control them. For ease of use, it is recommended that you use the hex2dec function to specify the bit mask in hexadecimal.

For example, to allow any length, enter

# Condor PCI-1553 RT Initialize

---

```
hex2dec( 'ffffffff' )
```

To allow a single message length, such as 5, enter

```
hex2dec( '00000010' )
```

because  $2^{(5-1)}=0x10$ . To allow message length N, this value should be  $2^{(N-1)}$ .

The board does not accept a message with a disallowed length and will send an error to the Bus Controller by returning a status with the Message Error bit set.

## Receive sub addresses

Enter a vector of the subaddresses that can accept a receive message. Each element of the vector must correspond to the corresponding element of the **Legal receive message lengths** vector.

A Remote Terminal only accepts a message if it has been directed to accept messages of the same length. During initialization, the board must receive a list of valid message lengths for each Remote Terminal number. To enable the subaddress for the Remote Terminal, include the subaddress in the vector.

## Legal receive message lengths

Enter a vector of receive message lengths. This is a vector of bit masks where each bit corresponds to a single message length. Ensure the following:

- Each element of the vector must correspond to the corresponding element of the **Transmit sub addresses** vector.
- Each element of the vector must correspond to the corresponding element of the **Receive sub addresses** vector.
- To enable a receive message, each element of the vector must correspond to the corresponding element of the **Legal transmit message lengths**. Failure to do this prevents the subaddress on this Remote Terminal from responding.



Because a message can have from 1 to 32 values in it, the block uses a single 32-bit integer to control them. For ease of use, it is recommended that you use the hex2dec function to specify the bit mask in hexadecimal.

For example, to allow any length, enter

```
hex2dec('ffffffff')
```

To allow a single message length, such as 5, enter

```
hex2dec('00000010')
```

because  $2^{(5-1)}=0x10$ . To allow message length N, this value should be  $2^{(N-1)}$ .

The board does not accept a message with a disallowed length and will send an error to the Bus Controller by returning a status with the Message Error bit set.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

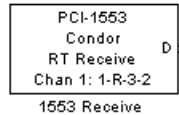
# Condor PCI-1553 RT Receive

---

**Purpose** Condor PCI-1553 Remote Terminal (RT) Receive

**Library** xPC Target Library for MIL-STD-1553

**Note** Your model might have multiple Remote Terminal Receive blocks. To help you locate a particular block, the Remote Terminal Receive block displays the number and address parameters.



The block display has the format

remote terminal-R-subaddress-number of words

Remote terminal — Is the **Remote Terminal** parameter

R — Indicates receive command

Subaddress — Is the **Sub address** parameter

Number of words — Is the **Number of words to receive** parameter

A receive command for RT 1, sub address 3 to receive 2 words has the layout 1-R-3-2.

## Block Parameters

### Channel

From the list, select from 1 or 2. This is the channel for this command stream. Select this value from the available channels on the board.

### Remote Terminal

From the list, choose a Remote Terminal from 1 to 31. Select the Remote Terminal number for this message.

### Sub address

From the list, choose a subaddress from 1 to 31. Select the sub address for this message.

## **Number of words to receive**

Enter the number of 16 bit integers to receive as the data part of this message. This number must be between 1 and 32. The output data vector will have this same vector width.

Note that if this Remote Terminal or subaddress might be sent messages with different lengths, specify the longest message length for this parameter. You must determine how much is significant from content or some other means.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Condor PCI-1553 RT Send

---

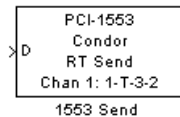
**Purpose** Condor PCI-1553 Remote Terminal (RT) Send block

**Library** xPC Target Library for MIL-STD-1553

**Description** A Remote Terminal sends data only if it receives a transmit command from the Bus Controller. This block prepares the board for the next transmit command on this channel, Remote Terminal number, and subaddress.

The data vector signal input to this block is copied to the board message buffer that corresponds to the specified address. The input data vector must have the same length as specified in the **Number of words to send** parameter. The input vector can be either 16- or 32-bit signed or unsigned data. The block only sends the bottom 16 bits of each element.

Your model might have multiple Remote Terminal Send blocks. To help you locate a particular block, the Remote Terminal Send block displays the number and address parameters.



The block display has the format

```
remote terminal-T-subaddress-number of words
```

Remote terminal — Derives from the **Remote Terminal** parameter

T — Indicates transmit (or send) command

Subaddress — Derives from the **Sub address** parameter

Number of words — Derives from the **Number of words to receive** parameter

A transmit command for RT 1, sub address 3 to send 2 words has the layout 1-T-3-2.

## **Block Parameters**

### **Channel**

From the list, select from 1 or 2. Ensure that you have initialized this channel with the PCI-1553 Initialize and RT Initialize blocks.

### **Remote Terminal**

From the list, choose a Remote Terminal from 1 to 31. Select the Remote Terminal number for this message.

### **Sub address**

From the list, choose a subaddress from 1 to 31. Select the sub address for this message.

### **Number of words to send**

Enter the number of 16 bit integers to send as the data part of this message. This number must be between 1 and 32. The input data vector must have the same vector width.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Condor QPCI-1553 Bus Controller Send

---

<b>Purpose</b>	Condor QPCI-1553 Bus Controller (BC) Send block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Description</b>	<p>This block sends a list of messages to the specified channel on the board. Before you construct the list of messages:</p> <ol style="list-style-type: none"><li>1 Allocate space for the list with the Condor 1553 Create BC Message List block.</li><li>2 Fill in information about each message with Condor 1553 Encode BC Message blocks.</li></ol> <p>Each instance of an Encode BC Message block fills in information for a single message on the list. To ensure that all Encode BC blocks execute before the Bus Controller Send block, daisy chain the Encode BC Message block L signals.</p> <p>The Bus Controller Send block allocates a list for the reception of messages. This block can</p> <ul style="list-style-type: none"><li>• Read the set of messages and statuses from the last time a message was sent before sending the new list.</li><li>• Send the new list and wait for a response.</li></ul> <p>In other words, for time step N, the output of the Send block is either the response to time step N-1 or the response to time step N, chosen from the <b>Response mode</b> parameter.</p>
<b>Block Parameters</b>	<b>Channel</b> From the list, choose 1, 2, 3, or 4. This is the channel for this command stream.

# Condor QPCI-1553 Bus Controller Send

---

## Response mode

Choose from

- **Read response to previous message then send new message** — Read the command buffers from the board before sending the new ones.
- **Send new message then wait and read response** — Transmit new message, then wait for response before continuing.

See the **Maximum wait** time parameter to set the amount of time the block should wait for a response.

## Maximum wait time (microseconds)

Enter the maximum time that this block waits for the message stream to be sent by the board. Use this parameter if the **Response mode** parameter is set to **Send new message then wait and read response**.

A reasonable maximum wait time is 1000 microseconds (1 millisecond). Note, this value should not exceed the sample time except for failsafe instances. Exceeding the sample time will trigger an execution overload.

## Sample time

Enter the base sample time or a multiple of the base sample time.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

## Condor QPCI-1553 Bus Controller Send

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



**Purpose** Condor QPCI-1553 Bus Monitor block

**Library** xPC Target Library for MIL-STD-1553

**Block** **Channel**

**Parameters**

From the list, choose 1, 2, 3, or 4.

**Maximum number of messages to receive**

Enter the maximum number of messages this block should read from the board.

**Sample time**

Enter the base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Condor QPCI-1553 Initialize

---

<b>Purpose</b>	Condor QPCI-1553 Initialize block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Block Parameters</b>	<p><b>Channel</b> From the list, select from 1, 2, 3, or 4. This is the channel to initialize on the board.</p> <p><b>Loopback enabled</b> Select this check box to route signals to the testbus on the board. This is a loopback connection between the four channels on the board without the need for external wiring. This is useful for testing.</p> <p><b>Initialize for Bus Controller operation</b> Select this check box to perform Bus Controller initialization for this channel. Selecting this check box enables the next four parameters.</p> <p><b>Initialize for Bus Monitor operation</b> Select this check box to initialize this channel for Bus Monitor operation. You must select this check box to initialize the channel for Bus Monitor operation. Selecting this check box enables the parameters:</p> <ul style="list-style-type: none"><li>• <b>Monitor bus A</b> — Select this check box to monitor bus A. You can monitor either bus A, bus B, or both.</li><li>• <b>Monitor bus B</b> — Select this check box to monitor bus B. You can monitor either bus B, bus A, or both.</li><li>• <b>Number of monitor buffers to allocate</b> — Enter the number of at least as many buffers as you expect to see messages between calls to the Bus Monitor block. If more messages arrive than there are buffers, some messages will be lost and not monitored.</li></ul> <p><b>Initialize for Remote Terminal operation</b> Select this check box to prepare the board to operate as a Remote Terminal. If you select this check box, add the RT Initialize block</p>

to the model to initialize the Remote Terminal. Connect the RT Initialize block for the Remote Terminal to the S output of this block.

Selecting this check box enables the parameter:

**RT address 31 is Broadcast**

Select this check box to enable the board to see messages to Remote Terminal 31 as broadcast messages.

**Sample time**

Enter the base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Condor QPCI-1553 RT Initialize

---

**Purpose** Condor QPCI-1553 RT Initialize block

**Library** xPC Target Library for MIL-STD-1553

**Description** Add an RT Initialize block for each Remote Terminal address for which you want this board to accept commands. Because each of these blocks must execute after the global board initialization, connect the S input to either

- Main Condor QPCI-1553 Initialize block for this board
- Another Condor QPCI-1553 RT Initialize block that is connected to the main Condor QPCI-1553 Initialize block.

This connection carries the channel number and PCI slot information.

## **Block Parameters**

### **Remote Terminal**

From the list, choose a Remote Terminal from 1 to 31. Choose the terminal number that this initialization block is configuring.

### **Connect to bus A**

Select this check box to have this Remote Terminal listen on bus A. Choose both A and B if a message can come in on either bus.

### **Connect to bus B**

Select this check box to have this Remote Terminal listen on bus B. Choose both A and B if a message can come in on either bus.

### **Initial status**

Enter the initial value of the status before any commands are executed. The board maintains a status word that is sent to the Bus Controller after command execution.

### **Initial BIT word**

Enter the initial value of Built In Test (BIT) word. The Remote Terminal sends this word in response to the Transmit BIT Word mode code.

## **Inhibit terminal flag**

Select this check box to inhibit the Remote Terminal flag. This is the initial value of the Inhibit Terminal Flag bit. This can be changed from the Bus Controller with the Inhibit Terminal and Override Inhibit Terminal mode codes. The terminal bit is one of the hardware maintained bits in the status word.

## **Transmit sub addresses**

Enter a vector of subaddresses to which this Remote Terminal should respond. These are the subaddresses that can transmit data when requested. Each element of the vector must correspond to the corresponding element of the **Legal transmit message lengths** vector. To enable the subaddress for the Remote Terminal, include the subaddress in the vector.

A Remote Terminal only accepts a message if it has been directed to accept messages of the same length. During initialization, the board must receive a list of valid message lengths for each Remote Terminal number.

## **Legal transmit message lengths**

Enter a vector of transmit message lengths. This is a vector of bit masks where each bit corresponds to a single message length. Ensure the following:

- Each element of the vector must correspond to the corresponding element of the **Transmit sub addresses** vector.
- To enable a transmit message, each element of the vector must correspond to the corresponding element of the **Legal receive message lengths**. Failure to do this prevents the subaddress on this Remote Terminal from responding.

Because a message can have from 1 to 32 values in it, the block uses a single 32-bit integer to control them. For ease of use, it is recommended that you use the hex2dec function to specify the bit mask in hexadecimal.

For example, to allow any length, enter

# Condor QPCI-1553 RT Initialize

---

```
hex2dec('ffffffff')
```

To allow a single message length, such as 5, enter

```
hex2dec('00000010')
```

because  $2^{(5-1)}=0x10$ . To allow message length N, this value should be  $2^{(N-1)}$ .

The board does not accept a message with a disallowed length and will send an error to the Bus Controller by returning a status with the Message Error bit set.

## Receive sub addresses

Enter a vector of the subaddresses that can accept a receive message. Each element of the vector must correspond to the corresponding element of the **Legal receive message lengths** vector. To enable the subaddress for the Remote Terminal, include the subaddress in the vector.

A Remote Terminal only accepts a message if it has been directed to accept messages of the same length. During initialization, the board must receive a list of valid message lengths for each Remote Terminal number.

## Legal receive message lengths

Enter a vector of receive message lengths. This is a vector of bit masks where each bit corresponds to a single message length. Ensure the following:

- Each element of the vector must correspond to the corresponding element of the **Transmit sub addresses** vector.
- Each element of the vector must correspond to the corresponding element of the **Receive sub addresses** vector.
- To enable a receive message, each element of the vector must correspond to the corresponding element of the **Legal transmit message lengths**. Failure to do this prevents the subaddress on this Remote Terminal from responding.

- Because a message can have from 1 to 32 values in it, the block uses a single 32-bit integer to control them. For ease of use, it is recommended that you use the hex2dec function to specify the bit mask in hexadecimal.

For example, to allow any length, enter

```
hex2dec('ffffffff')
```

To allow a single message length, such as 5, enter

```
hex2dec('00000010')
```

because  $2^{(5-1)}=0x10$ . To allow message length N, this value should be  $2^{(N-1)}$ .

The board does not accept a message with a disallowed length and will send an error to the Bus Controller by returning a status with the Message Error bit set.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

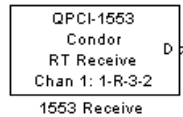
# Condor QPCI-1553 RT Receive

---

**Purpose** Condor QPCI-1553 Remote Terminal (RT) Receive block

**Library** xPC Target Library for MIL-STD-1553

**Note** Your model might have multiple Remote Terminal Receive blocks. To help you locate a particular block, the Remote Terminal Receive block displays the number and address parameters.



The block display has the format

remote terminal-R-subaddress-number of words

Remote terminal — Is the **Remote Terminal** parameter

R — Indicates receive command

Subaddress — Is the **Sub address** parameter

Number of words — Is the **Number of words to receive** parameter

A receive command for RT 1, sub address 3 to receive 2 words has the layout 1-R-3-2.

## Block Parameters

### Channel

From the list, select from 1 to 4. This is the channel for this command stream. Select this value from the available channels on the board.

### Remote Terminal

From the list, choose a Remote Terminal from 1 to 31. Select the Remote Terminal number for this message.

### Sub address

From the list, choose a subaddress from 1 to 31. Select the sub address for this message.



## **Number of words to receive**

Enter the number of 16 bit integers to receive as the data part of this message. This number must be between 1 and 32. The output data vector will have this same vector width.

Note that if this Remote Terminal or subaddress might be sent messages with different lengths, specify the longest message length for this parameter. You must determine how much is significant from content or some other means.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Condor QPCI-1553 RT Send

---

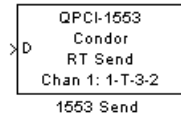
**Purpose** Condor QPCI-1553 Remote Terminal Send (RT) block

**Library** xPC Target Library for MIL-STD-1553

**Description** A Remote Terminal sends data only if it receives a transmit command from the Bus Controller. This block prepares the board for the next transmit command on this channel, Remote Terminal number, and subaddress.

The data vector signal input to this block is copied to the board message buffer that corresponds to the specified address. The input data vector must have the same length as specified in the **Number of words to send** parameter. The input vector can be either 16- or 32-bit signed or unsigned data. The block only sends the bottom 16 bits of each element.

Your model might have multiple Remote Terminal Send blocks. To help you locate a particular block, the Remote Terminal Send block displays the number and address parameters.



The block display has the format

remote terminal-T-subaddress-number of words

Remote terminal — Is the **Remote Terminal** parameter

T — Indicates transmit (or send) command

Subaddress — Is the **Sub address** parameter

Number of words — Is the **Number of words to receive** parameter

A transmit command for RT 1, sub address 3 to send 2 words has the layout 1-T-3-2.

## Block Parameters

### Channel

From the list, select from 1 to 4. Ensure that you have initialized this channel with the PCI-1553 Initialize and RT Initialize blocks.

### Remote Terminal

From the list, choose a Remote Terminal from 1 to 31. Select the Remote Terminal number for this message.

### Sub address

From the list, choose a subaddress from 1 to 31. Select the sub address for this message.

### Number of words to send

Enter the number of 16 bit integers to send as the data part of this message. This number must be between 1 and 32. The input data vector must have the same vector width.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# Access IO

---

This chapter describes the I/O board supported by xPC Target (<http://www.accessio.com>).

Access I/O WDG-CSM Watchdog  
Timer

The WDG-CSM is a watchdog timer  
used to detect computer failure.

## **Boards and Blocks**

**Board** Access I/O WDG-CSM

**General Description** The WDG-CSM is a watchdog timer used to detect computer failure. You can program this watchdog to reboot the system when a programmable timeout occurs. The timeout interval can range from 20 microseconds to 400 seconds.

xPC Target supports this board with one driver block:

- Access I/O WDG-CSM Watchdog Timer

<b>Board Characteristics</b>	Board name	WDG-CSM
	Manufacturer	Access IO
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# Access I/O WDG-CSM Watchdog Timer

---

<b>Purpose</b>	WDG-CSM Watchdog Timer block
<b>Library</b>	xPC Target Library for Access
<b>Block Parameters</b>	<p><b>Watchdog Time [s] (20us-4800s)</b> Enter a timeout value in seconds.</p> <p><b>Show reset port</b> Select this check box to enable an input port on the driver block. A signal connected to this port resets the watchdog.</p> <p><b>Sample time</b> Base sample time or a multiple of the base sample time.</p> <p><b>Base address</b> Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter  0x300</p>



# ADDI-DATA

---

This chapter describes I/O boards supported by xPC Target (<http://www.addi-data.de>).

ADDI-DATA APCI-1710

The APCI-1710 is a general-purpose counting board with four function modules.

ADDI-DATA PA-1700

The PA1700 is a counter board with three 24-bit counters for connecting three incremental encoders.

“Boards and Blocks — Alphabetical List” on page 10-2

Description of block parameters for ADDI-DATA driver blocks.

## **Boards and Blocks – Alphabetical List**

**Board** APCI-1710

**General Description** The APCI-1710 is a general-purpose counting board with four function modules.

xPC Target supports this board with this driver block:

- ADDI-DATA APCI-1710 Incremental Encoder

Note, xPC Target only supports the 32-bit mode for this board. As a result, each port only supports a single encoder.

<b>Board Characteristics</b>	Board name	APCI-1710
	Manufacturer	ADDI-DATA
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# ADDI-DATA APCI-1710 Incremental Encoder

---

**Purpose** APCI-1710 Incremental Encoder

**Library** xPC Target Library for ADDI-DATA

**Note** A function module is individually programmable with different firmware. You do this by using the ADDI-DATA utility SET1710. This driver supports the APCI-1710 if the specified function module is programmed with the incremental encoder firmware.

If the board and its specific module are not programmed with the incremental encoder firmware, you must invoke SET1710 before the driver can be used within an xPC Target application. In this case, plug the board into a PC running Microsoft Windows and install the board as indicated in the ADDI-DATA user manual. Use SET1710 to download the incremental encoder firmware onto the appropriate function module. After this step, you can remove the board and plug it into the target PC.

This driver block has two block outputs. The values output depend on the value of the **Type of Evaluation** parameter. See below for further information. Refer to the APCI-1710-manual for information on how to connect the encoders to the board.

## Block Parameters

### Function module

From the list select 1, 2, 3, or 4. This field specifies the function module (counter) to be used for this block. It must be programmed with the incremental encoder firmware. Two blocks for the same board cannot have the same module (channel) specified.

### Type of evaluation

From the list select the type of counter evaluation as either

- **Virtual Absolute** — Gets the counter value as an absolute value after the reference point of the encoder has been reached for the first time. The first output of the block outputs the absolute angle of the connected encoder in radians. As long as the reference point has not been reached for the first time, the second block output is zero. If the reference point is reached for the first time, and only for the first time, the corresponding

counter is reset to zero and the second output goes to 1. From then on the output 1 outputs an absolute angle even if the encoder is turned multiple times. The second output can be used for controlling or switching different Simulink submodels.

- **Reset and Index Output Up-Dating** — Gets the counter value in the range of  $0..2*\pi$  or  $-\pi..+\pi$  where the counter is reset every time the reference point is reached. The first output of the block outputs the angle of the connected encoder in radian. As long as the reference point has not been reached for the first time, the second block output is zero. Every time the reference point is reached, the counter is reset to zero and, depending on the direction of the encoder at this event, the output value is either incremented or decremented by the value 1. In other words the second output outputs the actual number of turns  $n$  because the reference point has been reached for the first time. If the second output is multiplied by  $2*\pi$  and added to the value of the first output, you get an absolute multiturn angle, even if the counter is reset periodically.

## **Mode**

From the list, choose single, double, or quadruple. This parameter specifies the phase detection mode, that is, how many phase changes of the specified module are detected (see the APC1-1710 manual).

## **Hysteresis**

From the list choose either off or on. The **Hysteresis** parameter specifies whether a counter should skip a tick if the direction changes (see the APC1-1700 manual).

## **Resolution**

Specifies the resolution of the connected incremental encoder for one revolution.

## **Sample time**

Model base sample time or a multiple of the base sample time.

# ADDI-DATA APCI-1710 Incremental Encoder

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board

The PA1700 is a counter board with three 24-bit counters for connecting three incremental encoders.

xPC Target supports this board with this driver block:

- ADDI-DATA PA-1700 Incremental Encoder

## Board Characteristics

Board name	PA1700
Manufacturer	ADDI-DATA
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# ADDI-DATA PA-1700 Incremental Encoder

---

**Purpose** PA-1700 Incremental Encoder block

**Library** xPC Target Library for ADDI-DATA

**Note** The driver block has two block outputs. The first outputs the absolute angle in radians. The second output is zero as long as the index or the reference point was not reached by rotating the encoder. If it is reached for the first time, and only for the first time, the corresponding counter is reset to zero and this output goes to 1. From then on the output 1 outputs an absolute angle even if the encoder is turned multiple times. The second output can be used for controlling or switching different Simulink submodels.

## Block Parameters

### Counter

From the list select 1, 2, or 3. This parameter specifies the counter used for this block. Two blocks for the same board (same base address) cannot have the same counter (channel) specified.

### Mode

From the list select single, double, or quadruple. This parameter specifies the phase detection mode, that is, how many phase changes of the specified counter are detected (see the PA1700 manual).

### Hysteresis

From the list choose either off or on. The Hysteresis parameter specifies whether a counter should skip a tick if the direction changes (see the PA1700 manual).

### Resolution

Specifies the resolution of the connected incremental encoder for one revolution.

### Sample time

Model base sample time or a multiple of the base sample time.



# ADDI-DATA PA-1700 Incremental Encoder

---

## Base Address

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

The following jumpers must be set according to the parameters entered above:

- Jumper J16, 17, and 18 must be set to position 1-2.
- Jumper J13, 14, and 15 must be set to position 1-2.
- Jumper J1, 5, and 9 must be set according to the connected encoders.
- Jumper J2, 6, and 10 must be set according to the connected encoders.
- Jumper J3, 7, and 11 must be set according to the connected encoders.
- Jumper J4, 8, and 12 must be set according to the connected encoders.

For information on how to connect the encoders to the board, see the PA1700 manual.

If you want to use the 5 V power supply from the board (PIN20), you must insert Fuse 1 on the board. Refer to the PA1700 manual.



# Adlink

---

This chapter describes the Adlink boards supported by xPC Target (<http://www.adlinktech.com>).

Adlink PCI-8133

Three-phase encoder counter and PWM output board.

Adlink PCI-6208

Digital-to-analog converter board with four bits of digital input and four bits of digital output lines.

“Boards and Blocks — Alphabetical List” on page 11-2

Description of block parameters for Adlink driver blocks.

## **Boards and Blocks – Alphabetical List**

**Board** Adlink PCI-8133

**General Description** The Adlink PCI-8133 is a three-phase encoder counter and PWM output board. This board has three 16-bit quadruple AB phase encoder counters, 12-bit PWM resolution, and eight general-purpose digital input and output lines.

xPC Target supports the three-phase PWM generation section of the board with this driver block:

- Adlink PCI-8133 3-Phase PWM

<b>Board Characteristics</b>	Board name	PCI-8133
	Manufacturer	Adlink
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	No

# Adlink PCI-8133 3-Phase PWM

---

**Purpose** Adlink PCI-8133 3-Phase pulse width modulator block

**Library** xPC Target Library for Adlink

<b>Scaling Output to Input</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Duty cycle: double	0 to 1

- Note**
- There is one input port for each phase (channel). You can select and order each phase (channel) individually.
  - Hardware outputs are open collector lines that can draw a maximum current of 20 mA.
  - To enable PWM generation, ensure that the OENA pin (pin 34 of connector CN1) is connected to pin VCC (pin 19 of connector CN1).
  - Although the duty cycle inputs are of type double, the duty cycle resolution is finite. The value of the **Factor n determining square wave period** parameter defines the duty cycle resolution. For example, if the value of **Factor n determining square wave period** is 1000 for an output period of 200 microseconds, the duty cycle can be adapted with a resolution of 1000 steps (from 0...1), in relation to the value of **Factor n determining square wave period**. The duty cycle resolution lowers by a smaller output period (or higher output frequency).

## **Block Parameters**

### **Factor n determining square wave period**

Defines the period (duration) of the square wave, where the square wave is the sum of the on and off part. This parameter is also called the n factor. The n factor must be in the range from 1 to 65535. The resulting period is calculated as

$$T=n*200 \text{ nanoseconds}$$

## **Factor m determining dead time duration**

Defines the duration of the dead time that is needed if the output lines drive transistor bridges. This parameter is also called the m factor. The m factor must be in the range from 1 to 255. The resulting duration is calculated as

$$T_{dt} = 750 \text{ nanoseconds} * (m+1)$$

## **Channel vector**

Defines the channel (phase) that is active. Enter a vector of numbers between 1 and 3 to This parameter also specifies the input port of the block is connected to the channel. Channel value 1 represents phase U, value 2 represents phase V, and value 3 represents phase W. The maximum length of the vector is 3.

## **Reset vector**

Enter a scalar or a vector that is the same length and channel order as the **Channel vector** value. Enter 1 or 0. A value of 1 resets the output. A value of 0 retains the last value of the duty cycle when the target application stops. You can specify a different reset vector value for each channel.

## **Initial duty cycle vector**

Enter a scalar or a vector that is the same length and channel order as the **Channel vector** value. Enter 1 or 0. A value of 1 sets the reset duty cycle for the corresponding channel if the **Reset vector** for that channel is also 1.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# Adlink PCI-8133 3-Phase PWM

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



**Board** Adlink PCI-6208

**General Description**

The Adlink PCI-6208A is a digital to analog converter board. This board has four bits of digital input and four bits of digital output lines.

xPC Target supports this board with the following driver blocks:

- Adlink PCI-6208A Analog Output
- Adlink PCI-6208A Digital Input
- Adlink PCI-6208A Digital Output

**Board Characteristics**

Board name	PCI-6208A
Manufacturer	Adlink
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Adlink PCI-6208A Analog Output

---

**Purpose** Adlink PCI-6208A Analog Output block

**Library** xPC Target Library for Adlink

<b>Scaling Output to Input</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	ma	Double	1

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the output channels.

For example, to use the first and second analog output (D/A) channels enter

[1, 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4.

### **Range**

From the list, choose

- 0 to 20 ma
- 5 to 25 ma
- 4 to 20 ma

Signal input to the block is in milliamps (ma). The output voltage maximum is 10 volts. If the resistance is too large to get the chosen current, the voltage measured across the output load will saturate.

For example, assume an input sine wave with:

$$\text{Input} = 10 \cdot \sin(\omega \cdot t) + 10$$

If you set the range to 0 to 20 ma, the current should oscillate from 0 to 20 ma at  $\omega$  radians per second. The current will only reach 20 ma if the load resistance is less than or equal to 500 ohms. If the load resistance is 1000 ohms, the maximum current that 10 volts can drive is only 10 ma. The top half of the sine wave will be clipped off.

Note the following:

- The scaling in this block is correct for the current output ports, not the voltage ones.
- This block limits output to positive values only. This is in response to the board hardware manual caution that states driving a negative voltage from the D/A converter to the voltage to current output board might cause damage to the current output.

### **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. For example, if **Channel vector** is [1 2] and the **Reset vector** is [1], the action taken will be the same as if Reset vector was set to [1 1]. Both channels will be reset to their initial values when model execution is stopped.

### **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. When model execution is stopped, the

# Adlink PCI-6208A Analog Output

---

corresponding position in **Reset vector** is checked. Depending on that value, the channel is either reset to the initial value or remains at the last value attained while the model was running. For example, assume that **Channel vector** is [ 1 2 ], **Reset vector** is [ 1 0 ], and **Initial value vector** is [2.3 5.6]. On initial download, channel 1 is set to 2.3 ma and channel 2 to 5.6 ma. When the model is stopped, channel 1 resets to 2.3 ma and channel 2 remains at the last value attained.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Adlink PCI-6208A Digital Input

**Purpose** Adlink PCI-6208A Digital Input

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low = 0 TTL high = 1

## Block Parameters

### Channel vector

Enter a vector of numbers to specify the digital input port ports.

For example, to use the first and second digital input channels enter

```
[1, 2]
```

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4. Each input can be listed at most once in this vector.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Adlink PCI-6208A Digital Output

---

**Purpose** Adlink PCI-6208A Digital Output

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the digital output channels.

For example, to use the first and second analog digital output channels enter

[1, 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 8.

### **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. For example, if **Channel vector** is [1 2] and the **Reset vector** is [1], the action taken will be the same as if Reset vector was set to [1 1]. Both channels will be reset to their initial values when model execution is stopped.

### **Initial value vector**

The initial value vector contains the initial voltage values for the digital 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. When model execution is stopped, the corresponding position in **Reset vector** is checked. Depending on that value, the channel is either reset to the initial value or remains at the last value attained while the model was running. For example, assume that **Channel vector** is [ 1 2 ], **Reset vector** is [ 1 0], and **Initial value vector** is [ 0 1]. On initial download, channel 1 is set to off and channel 2 to on. When the model is stopped, channel 1 resets to off and channel 2 remains at the last value attained.

If the **Initial value vector** value is greater than 0.5, the corresponding digital output port is turned on, otherwise it is turned off.

### **Sample time**

Base sample time or a multiple of the base sample time.

### **PCI slot**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```





# Advantech

---

This chapter describes I/O boards supported by xPC Target (<http://www.advantech.com>).

Advantech PCL-1800	16 single or 8 differential analog channels, 2 analog output D/A channels, 16 digital input lines, and 16 digital output lines.
Advantech PCL-711B	Eight single-ended analog input channels, 1 analog output channel, 16 digital input lines, and 16 digital output lines.
Advantech PCL-726	Six independent analog output D/A channels, 16 digital input lines, and 16 digital output lines.
Advantech PCL-727	12 independent analog output D/A channels, 16 digital input lines, and 16 digital output lines.
Advantech PCL-728	Two independent analog output D/A channels.
Advantech PCL-812	16 single-ended analog input channels, 2 analog output D/A channels, 16 digital input lines, and 16 digital output lines.
Advantech PCL-812PG	16 single or 8 differential analog channels, 2 analog output D/A channels, 16 digital input lines, and 16 digital output lines.
Advantech PCL-818	16 single or 8 differential analog channels, 2 analog output D/A channels, 16 digital input lines, and 16 digital output lines.

Advantech PCL-818H	16 single or 8 differential analog channels, 1 analog output D/A channel, 16 digital input lines, and 16 digital output lines.
Advantech PCL-818HD	16 single or 8 differential analog channels, 1 analog output D/A channel, 16 digital input lines, and 16 digital output lines.
Advantech PCL-818HG	16 single or 8 differential analog input (A/D) channels, 1 analog output (D/A) channel, 16 digital input lines, and 16 digital output lines.
Advantech PCL-818L	16 single or 8 differential analog input (A/D) channels, 1 analog output (D/A) channel, 16 digital input lines, and 16 digital output lines.
“Boards and Blocks — Alphabetical List” on page 12-3	Description of block parameters for Advantech driver blocks.

## **Boards and Blocks – Alphabetical List**

# Advantech PCL-1800

---

## Board

Advantech PCL-1800

## General Description

The PCL-1800 is an I/O board with 16 single or eight differential analog channels (12-bit) with a maximum sample rate of 330 kHz, two analog output D/A channels (12-bit), and 16 digital input lines and 16 digital output lines.

xPC Target supports this board with these driver blocks:

- Advantech PCL-1800 Analog Input (A/D)
- Advantech PCL-1800 Analog Output (D/A)
- Advantech PCL-1800 Digital Input
- Advantech PCL-1800 Digital Output

## Board Characteristics

Board Name	PCL-1800
Manufacturer	Advantech
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Advantech PCL-1800 Analog Input (A/D)

**Purpose** PCL-1800 Analog Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

If you select 16 channels (**Input coupling** parameter set to Single-ended), enter numbers between 1 and 16. If you select differential (**Input coupling** parameter set to differential), enter numbers between 1 and 8. For example, to use the first and second analog output (A/D) channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code	Input Range (V)	Range Code
-10 to +10	-10	0 to +10	10
-5 to +5	-5	0 to +5	5
-2.5 to +2.5	-2.5	0 to +2.5	2.5

# Advantech PCL-1800 Analog Input (A/D)

---

Input Range (V)	Range Code	Input Range (V)	Range Code
-1.25 to +1.25	-1.25	0 to +1.25	1.25
-0.625 to +0.625	-0.625		

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board.

## Input coupling

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- differential channels (8 channels)

Your choice must correspond to the MUX switch setting on the board.

## 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

# Advantech PCL-1800 Analog Output (D/A)

**Purpose** PCL-1800 Analog Output block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>	<b>Input Range (V)</b>	<b>Range Code</b>
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]

# Advantech PCL-1800 Analog Output (D/A)

---

The range settings must correspond to the DIP switch settings on the board.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



# Advantech PCL-1800 Digital Input

**Purpose** PCL-1800 Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital inputs, 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
```

# Advantech PCL-1800 Digital Output

---

**Purpose** PCL-1800 Digital Output

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital outputs, 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

**Board** Advantech PCL-711B

**General Description**

The PCL-711B is an I/O board with eight single-ended analog input channels (12-bit) with a maximum sample rate of 25 kHz, one analog output channel (12-bit), and 16 digital input lines and 16 digital output lines.

xPC Target supports this board with four driver blocks:

- Advantech PCL-711B Analog Input (A/D)
- Advantech PCL-711B Analog Output (D/A)
- Advantech PCL-711B Digital Input
- Advantech PCL-711B Digital Output

**Board Characteristics**

Board name	PCL-711B
Manufacturer	Advantech
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Advantech PCL-711B Analog Input (A/D)

**Purpose** PCL-711B Analog Input block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8. For example, to use the first and second analog output (A/D) channels, enter

[1,2]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.25 to +1.25	-1.25
-0.625 to +0.625	-0.625

For example, if the first channel is -10 to +10 volts and the second channel is -2.5 to +2.5 volts, enter

# Advantech PCL-711B Analog Input (A/D)

---

[-10, -2.5]

The range settings must correspond to the DIP switch settings on the board.

## **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

# Advantech PCL-711B Analog Output (D/A)

---

**Purpose** PCL-711B Analog Output block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

**Block Parameters**

**Range**

From the list, choose either 0-10V or 0-5V.

The range settings must correspond to the DIP switch settings on the board.

**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

# Advantech PCL-711B Digital Input

**Purpose** PCL-711B Digital Input

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. 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 the first eight digital inputs, 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
```

# Advantech PCL-711B Digital Output

---

**Purpose** PCL-711B Digital Output block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. 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 the first eight digital outputs, 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



**Board** Advantech PCL-726

**General Description** The PCL-726 is an I/O board with, six independent analog output D/A channels (12-bit), 16 digital input lines and 16 digital output lines.

xPC Target supports this board with these driver blocks:

- Advantech PCL-726 Analog Output (D/A)
- Advantech PCL-726 Digital Input
- Advantech PCL-726 Digital Output

<b>Board Characteristics</b>	Board name	PCL-726
	Manufacturer	Advantech
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Advantech PCL-726 Analog Output (D/A)

---

**Purpose** PCL-726 Analog Output block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 6. 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 vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Advantech PCL-726 Analog Output (D/A)

---

For example, if the first channel is 0 to +10 volts and the second channel is 0 to +5 volts, enter

[10,5]

The range settings must correspond to the DIP switch settings on the board.

## **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

# Advantech PCL-726 Digital Input

---

**Purpose** PCL-726 Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital inputs, 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

# Advantech PCL-726 Digital Output

**Purpose** PCL-726 Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital outputs, 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
```

# Advantech PCL-727

---

**Board** Advantech PCL-727

**General Description** The PCL-727 is an I/O board with 12 independent analog output D/A channels (12-bit), 16 digital input lines and 16 digital output lines.

xPC Target supports this board with these driver blocks:

- Advantech PCL-727 Analog Output (D/A)
- Advantech PCL-727 Digital Input
- Advantech PCL-727 Digital Output

<b>Board Characteristics</b>	Board name	PCL-727
	Manufacturer	Advantech
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Advantech PCL-727 Analog Output (D/A)

**Purpose** PCL-727 Analog Output block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 12. 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 vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Advantech PCL-727 Analog Output (D/A)

---

For example, if the first channel is 0 to +10 volts and the second channel is 0 to +5 volts, enter

[10,5]

The range settings must correspond to the DIP switch settings on the board.

## **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



# Advantech PCL-727 Digital Input

**Purpose** PCL-727 Digital Input block

**Library** xPC Target Library for Advantech

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital inputs, 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
```

# Advantech PCL-727 Digital Output

---

**Purpose** PCL-727 Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

<0.5 = TTL low

≥0.5 = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital outputs, 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

**Board** Advantech PCL-728

**General Description**

The PCL-728 is an I/O board with two independent analog output D/A channels (12-bit).

xPC Target supports this board with this driver block:

- Advantech PCL-728 Analog Output (D/A)

**Board Characteristics**

Board name	PCL-728
Manufacturer	Advantech
Bus Type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Advantech PCL-728 Analog Output (D/A)

**Purpose** PCL-728 Analog Output block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 2. 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]

Channel numbers begin with 1 even if the board manufacturer starts numbering channels with 0.

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Advantech PCL-728 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board.

## **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

# Advantech PCL-812

---

## Board

Advantech PCL-812

## General Description

The PCL-812 is an I/O board with 16 single-ended analog input channels (12-bit) with a maximum sample rate of 30 kHz, two analog output D/A channels (12-bit), and 16 digital input lines and 16 digital output lines.

xPC Target supports this board with these driver blocks:

- Advantech PCL-812 Analog Input (A/D)
- Advantech PCL-812 Analog Output (D/A)
- Advantech PCL-812 Digital Input
- Advantech PCL-812 Digital Output

## Board Characteristics

Board name	PCL-812
Manufacturer	Advantech
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Advantech PCL-812 Analog Input (A/D)

**Purpose** PCL-812 Analog Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. For example, to use the first and second analog input (A/D) channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.0 to +2.0	-2.0
-1.0 to +1.0	-1.25

# Advantech PCL-812 Analog Input (A/D)

---

For example, if the first channel is -10 to +10 volts and the second channel is -5 to +5 volts, enter

[ -10, -5 ]

The range settings must correspond to the DIP switch settings on the board.

## **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



# Advantech PCL-812 Analog Output (D/A)

**Purpose** PCL-812 Analog Output block

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range code
0 to +5	5

For example, if both channels are 0 to +5 volts, enter

[5,5]

The range settings must correspond to the DIP switch settings on the board.

# Advantech PCL-812 Analog Output (D/A)

---

## **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

# Advantech PCL-812 Digital Input

**Purpose** PCL-812 Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. 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 the first eight digital inputs, 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
```

# Advantech PCL-812 Digital Output

---

**Purpose** PCL-812 Digital Output block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. 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 the first eight digital outputs, 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

## Board

Advantech PCL-812PG

## General Description

The PCL-812PG is an I/O board with 16 single or eight differential analog channels (12-bit) with a maximum sample rate of 30 kHz, two analog output D/A channels (12-bit), and 16 digital input lines and 16 digital output lines.

xPC Target supports this board with these driver blocks:

- Advantech PCL-812PG Analog Input (A/D)
- Advantech PCL-812PG Analog Output (D/A)
- Advantech PCL-812PG Digital Input
- Advantech PCL-812PG Digital Output

## Board Characteristics

Board name	PCL-812PG
Manufacturer	Advantech
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Advantech PCL-812PG Analog Input (A/D)

**Purpose** PCI-812PG Analog Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. For example, to use the first and second analog input (A/D) channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.25 to +1.25	-1.25
-0.625 to +0.625	-0.625
0 to +10	10

# Advantech PCL-812PG Analog Input (A/D)

---

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5
0 to +1.25	1.25

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board.

## **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

# Advantech PCL-812PG Analog Output (D/A)

**Purpose** PCL-812PG Analog Output block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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



# Advantech PCL-812PG Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board.

## **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

# Advantech PCL-812PG Digital Input

---

**Purpose** PCL-812PG Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. 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 the first eight digital inputs, 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

# Advantech PCL-812PG Digital Output

**Purpose** PCL-812PG Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. 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 the first eight digital outputs, 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
```

# Advantech PCL-818

---

## Board

Advantech PCL-818

## General Description

The PCL-818 is an I/O board with 16 single or eight differential analog channels (12-bit) with a maximum sample rate of 100 kHz, two analog output D/A channels (12-bit), and 16 digital input lines and 16 digital output lines.

xPC Target supports this board with these driver blocks:

- Advantech PCL-818 Analog Input (A/D)
- Advantech PCL-818 Analog Output (D/A)
- Advantech PCL-818 Digital Input
- Advantech PCL-818 Digital Output

## Board Characteristics

Board name	PCL-818
Manufacturer	Advantech
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Advantech PCL-818 Analog Input (A/D)

**Purpose** PCL-818 Analog Input (A/D)

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## Block Parameters

### Channel vector

If you select 16 channels (**Input coupling** parameter set to Single-ended), enter numbers between 1 and 16. If you select differential (**Input coupling** parameter set to differential), enter numbers between 1 and 8. For example, to use the first and second analog output (A/D) channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.25 to +1.25	-1.25

# Advantech PCL-818 Analog Input (A/D)

---

Input Range (V)	Range Code
-0.625 to +0.625	-0.625
0 to 10	10
0 to +5	5
0 to +2.5	2.5
0 to +1.25	1.25

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board.

## Input coupling

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- differential channels (8 channels)

Your choice must correspond to the MUX switch setting on the board.

## 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

# Advantech PCL-818 Analog Output (D/A)

**Purpose** PCL-818 Analog Output block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 2. 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 vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Advantech PCL-818 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board.

## **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



# Advantech PCL-818 Digital Input

**Purpose** PCL-818 Digital Input

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. 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 the first eight digital inputs, 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
```

# Advantech PCL-818 Digital Output

---

**Purpose** PCL-818 Digital Output

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. 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 the first eight digital outputs, 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

**Board** Advantech PCL-818H

**General Description** The PCL-818H is an I/O board with 16 single or eight differential analog channels (12-bit) with a maximum sample rate of 100 kHz, one analog output D/A channel (12-bit), and 16 digital input lines and 16 digital output lines.

xPC Target supports this board with these driver blocks:

<b>Board Characteristics</b>	Board name	PCL-818H
	Manufacturer	Advantech
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Advantech PCL-818H Analog Input (A/D)

**Purpose** PCL-818H Analog Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

If you select 16 channels (**Input coupling** parameter set to Single-ended), enter numbers between 1 and 16. If you select differential (**Input coupling** parameter set to differential), enter numbers between 1 and 8. For example, to use the first and second analog output (A/D) channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.25 to +1.25	-1.25

# Advantech PCL-818H Analog Input (A/D)

Input Range (V)	Range Code
-0.625 to +0.625	-0.625
0 to 10	10
0 to +5	5
0 to +2.5	2.5
0 to +1.25	1.25

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board.

## Input coupling

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- differential channels (8 channels)

This choice must correspond to the MUX switch setting on the board.

## 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

# Advantech PCL-818H Analog Output (D/A)

---

**Purpose** PCL-818H Analog Output block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Range

From the list, choose either 0-10V or 0-5V.

The range settings must correspond to the DIP switch settings on the board.

### 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

# Advantech PCL-818H Digital Input

**Purpose** PCL-818H Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital inputs, 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
```

# Advantech PCL-818H Digital Output

---

**Purpose** PCL-818H Digital Output block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital outputs, 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



## Board

Advantech PCL-818HD

## General Description

The PCL-818HD is an I/O board with 16 single or eight differential analog channels (12-bit) with a maximum sample rate of 100 kHz, one analog output D/A channels (12-bit), and 16 digital input lines and 16 digital output lines.

xPC Target supports this board with these driver blocks:

- Advantech PCL-818HD Analog Input (A/D)
- Advantech PCL-818HD Analog Output (D/A)
- Advantech PCL-818HD Digital Input
- Advantech PCL-818HD Digital Output

## Board Characteristics

Board name	PCL-818HD
Manufacturer	Advantech
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Advantech PCL-818HD Analog Input (A/D)

**Purpose** PCL-818HD Analog Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

If you select 16 channels (**Input coupling** parameter set to Single-ended), enter numbers between 1 and 16. If you select differential (**Input coupling** parameter set to differential), enter numbers between 1 and 8. For example, to use the first and second analog output (A/D) channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.25 to +1.25	-1.25

# Advantech PCL-818HD Analog Input (A/D)

Input Range (V)	Range Code
-0.625 to +0.625	-0.625
0 to 10	10
0 to +5	5
0 to +2.5	2.5
0 to +1.25	1.25

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board.

## Input coupling

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- differential channels (8 channels)

Your choice must correspond to the MUX switch setting on the board.

## 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

# Advantech PCL-818HD Analog Output (D/A)

---

**Purpose** PCL-818HD Analog Output block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Range**

From the list, choose either 0 - 10V or 0 - 5V.

The range settings must correspond to the DIP switch settings on the board.

### **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

# Advantech PCL-818HD Digital Input

**Purpose** PCL-818HD Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital inputs, 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
```

# Advantech PCL-818HD Digital Output

---

**Purpose** PCL-818HD Digital Input block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital outputs, 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

## Board

Advantech PCL-818HG

## General Description

The PCL-818 is an I/O board with 16 single or eight differential analog input (A/D) channels (12-bit) with a maximum sample rate of 100 kHz, one analog output (D/A) channel (12-bit), and 16 digital input lines and 16 digital output lines.

xPC Target supports this board with these driver blocks:

- Advantech PCL-818HG Analog Input (A/D)
- Advantech PCL-818HD Analog Output (D/A)
- Advantech PCL-818HG Digital Input
- Advantech PCL-818HG Digital Output

## Board Characteristics

Board name	PCL-818HG
Manufacturer	Advantech
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Advantech PCL-818HG Analog Input (A/D)

**Purpose** PCL-818HG Analog Input block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

If you select 16 channels (**Input coupling** parameter set to Single-ended), enter numbers between 1 and 16. If you select differential (**Input coupling** parameter set to differential), enter numbers between 1 and 8. For example, to use the first and second analog output (A/D) channels, enter

[1,2]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.25 to +1.25	-1.25
-0.625 to +0.625	-0.625



# Advantech PCL-818HG Analog Input (A/D)

Input Range (V)	Range Code
0 to +10	10
0 to +1	1
0 to +0.1	0.1
0 to +0.01	0.01

For example, if the first channel is -10 to +10 volts and the second channel is -5 to +5 volts, enter

[ -10, -5]

The range settings must correspond to the DIP switch settings on the board.

## Input coupling

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- differential channels (8 channels)

Your choice must correspond to the MUX switch setting on the board.

## 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

# Advantech PCL-818HG Analog Output (D/A)

---

**Purpose** PCL-818HG Analog Output block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

**Block Parameters**

**Range**

From the list, choose either 0-10V or 0-5V.

The range settings must correspond to the DIP switch settings on the board.

**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

# Advantech PCL-818HG Digital Input

**Purpose** PCL-818HG Digital Input block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital inputs, 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
```

# Advantech PCL-818HG Digital Output

---

**Purpose** PCL-818HG Digital Output block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital outputs, 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

**Board** Advantech PCL-818L

**General Description**

The PCL-818L is an I/O board with 16 single or eight differential analog input (A/D) channels (12-bit) with a maximum sample rate of 40 kHz, one analog output (D/A) channels (12-bit), 16 digital input lines, and 16 digital output lines.

xPC Target supports this board with these driver blocks:

- Advantech PCL-818L Analog Input (A/D)
- Advantech PCL-818L Analog Output (D/A)
- Advantech PCL-818L Digital Input
- Advantech PCL-818L Digital Output

**Board Characteristics**

Board name	PCL-818
Manufacturer	Advantech
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Advantech PCL-818L Analog Input (A/D)

**Purpose** PCL-818L Analog Input block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

If you select 16 channels (**Input coupling** parameter set to Single-ended), enter numbers between 1 and 16. If you select differential (**Input coupling** parameter set to differential), enter numbers between 1 and 8. For example, to use the first and second analog output (A/D) channels, enter

[1,2]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.25 to +1.25	-1.25
-0.625 to +0.625	-0.625

# Advantech PCL-818L Analog Input (A/D)

---

For example, if the first channel is -2.5 to +2.5 volts and the second channel is -5 to +5 volts, enter

[ -2.5, -5 ]

The range settings must correspond to the DIP switch settings on the board.

If you need a -10 to +10 volt range, perform the following workaround. Note that the PCL-818L Analog Input block sets bits G1 and G0 in the board register, but cannot set or read the board jumper.

- 1** On the board, move the JP7 jumper to the 10 volt range (see the manufacturer documentation for details).
- 2** In the **Range vector** parameter, enter [ -5 ].

Setting the JP7 jumper to the 10 volt range doubles all input voltage ranges.

- 3** Use a Gain block to multiply the value by 2. This ensures that the output voltage matches the input voltage for all ranges.

## **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- differential channels (8 channels)

Your choice must correspond to the MUX switch setting on the board.

## **Sample time**

Base sample time or a multiple of the base sample time.

# Advantech PCL-818L Analog Input (A/D)

---

## **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



# Advantech PCL-818L Analog Output (D/A)

---

**Purpose** PCL-818L Analog Output block

**Library** xPC Target Library for Advantech

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

**Block Parameters**

**Range**

From the list, choose either 0-10V or 0-5V.

The range setting must correspond to the DIP switch settings on the board.

**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

# Advantech PCL-818L Digital Input

---

**Purpose** PCL-818L Digital Input block

**Library** xPC Target Library for Advantech

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

## **Block Parameters**

### **Channel vector**

Enter channels between 1 and 16. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital inputs, 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

# Advantech PCL-818L Digital Output

**Purpose** PCL-818L Digital Output block

**Library** xPC Target Library for Advantech

## Scaling of Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

<0.5 =  
TTL low  
≥0.5 = TTL  
high

## Block Parameters

### Channel vector

Enter channels between 1 and 16. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use the first eight digital outputs, 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



# Analogic

---

This chapter describes the Analogic I/O boards supported by xPC Target (<http://www.analogic.com>).

Analogic AIM	I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit)
Analogic AIM16	I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit)
“Boards and Blocks — Alphabetical List” on page 13-2	Description of block parameters for Analogic driver blocks.

## **Boards and Blocks – Alphabetical List**

## Board

Analogic AIM12

## General Description

This board comes in three configurations from the factory. The configurations are the AIM12-1/104, the AIM16-1/104, and the AIM16-2/104. The AIM16-1/104 and the AIM16-2/104 differ only in the time needed to acquire each sample. The AIM12-1/104 acquires 4 fewer bits, but since the 12 bits are placed in the high 12 bits of the result word, the internal scaling factors are identical to those used with the AIM16. The output of the AIM12 changes in larger steps. Gain settings for the AIM12 are also different than for the AIM16.

When acquiring samples from multiple channels, the board is driven in its burst mode when the highest clock rate available for the board is used. The AIM12-1/104 can acquire all 16 channels in 160 microseconds.

The AIM12 hardware can acquire from 1 to 16 (single ended) channels at each sample time. These channels must be in a contiguous group, but can start with any channel number and end with any channel number that is greater than or equal to the start channel number. A single channel is acquired by setting both first channel and last channel to the same value.

The choice of unipolar or bipolar conversion range is made by placing jumpers in accordance with the hardware manual. The driver is able to read the hardware configuration and adjusts the integer to float conversion as necessary.

xPC Target supports this board with these driver blocks:

- Analogic AIM12 Analog Input (A/D)
- Analogic AIM12 Digital Input
- Analogic AIM12 Digital Output

## Board Characteristics

Board name	AIM12
Manufacturer	Analogic
Bus type	PC/104
Access method	I/O mapped

# Analogic AIM

---

Multiple block instance support	No
Multiple board support	Yes



# Analogic AIM12 Analog Input (A/D)

**Purpose** Analogic AIM12 Analog Input block

**Library** xPC Target Library for Analogic

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

## **Block Parameters**

### **First channel**

In Single-ended (16 channels) input mode, enter a number between 1 and 16 to indicate the first channel to be acquired. In Differential (8 channels) input mode, enter a number between 1 and 8.

### **Last channel**

This is the last channel to acquire. The number must be greater than or equal to the first channel. In Single-ended (16 channels) input mode, the number must be less than or equal to 16. In Differential (8 channels) input mode, the number must be less than or equal to 8.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

Check the hardware manual of the board for wiring configurations.

### **Gain vector**

Enter a vector of gain values with one entry for each channel in the range first channel to last channel. Allowable gain settings for the AIM12 are 1, 10, and 100. If you enter a scalar for gain, this gain value is used for all channels.

# Analogic AIM12 Analog Input (A/D)

---

**Sample time**

Enter a base sample time or a multiple of the base sample time.

**Base address**

Enter the base address of the board in hexadecimal (such as 0x300) as determined by the hardware jumpers on the board.

**Purpose** AIM12 Digital Input block

**Library** xPC Target Library for Analogic

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 8 to select the digital input lines to be read. 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 of the digital inputs enter

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

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

### Channel group

Choose Lower 8 bits to connect to channels 1 through 8 or Upper 8 bits to connect to channels 9 through 16.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### Base address

Enter the base address of the board in hexadecimal (such as 0x300) as determined by the hardware jumpers on the board.

# Analogic AIM12 Digital Output

---

**Purpose** AIM12 Digital Output

**Library** xPC Target Library for Analogic

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital input lines to be read. 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 of the digital inputs enter

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

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

### **Channel group**

Choose Lower 8 bits to connect to channels 1 through 8 or Upper 8 bits to connect to channels 9 through 16.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **Base address**

Enter the base address of the board in hexadecimal (such as 0x300) as determined by the hardware jumpers on the board.

## Board

Analogic AIM16

## General Description

This board comes in three configurations from the factory. The configurations are the AIM12-1/104, the AIM16-1/104 and the AIM16-2/104. The AIM16-1/104 and the AIM16-2/104 differ only in the time needed to acquire each sample. The AIM12-1/104 acquires 4 fewer bits, but since the 12 bits are placed in the high 12 bits of the result word, the internal scaling factors are identical to those used with the AIM16. The output of the AIM12 changes in larger steps. Gain settings for the AIM12 are also different than for the AIM16.

When acquiring samples from multiple channels, the board is driven in its burst mode when the highest clock rate available for the board is used. The AIM16-1/104 can acquire all 16 channels in 160 microseconds while the AIM16-2/104 completes the same acquisition in 80 microseconds.

The AIM16 hardware can acquire from 1 to 16 (single ended) channels at each sample time. These channels must be in a contiguous group, but can start with any channel number and end with any channel number that is greater than or equal to the start channel number. A single channel is acquired by setting both first channel and last channel to the same value.

The choice of unipolar or bipolar conversion range is made by placing jumpers in accordance with the hardware manual. The driver is able to read the hardware configuration and adjusts the integer to float conversion as necessary.

xPC Target supports this board with these driver blocks:

- Analogic AIM16 Analog Input (A/D)
- Analogic AIM16 Digital Input
- Analogic AIM16 Digital Output

# Analogic AIM16

---

## **Board Characteristics**

Board name	AIM16
Manufacturer	Analogic
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Analog AIM16 Analog Input (A/D)

**Purpose** AIM16 Analog Input block

**Library** xPC Target Library for Analog

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

## Block Parameters

### First channel

In Single-ended (16 channels) input mode, enter a number between 1 and 16 to indicate the first channel to be acquired. In Differential (8 channels) input mode, enter a number between 1 and 8.

### Last channel

This is the last channel to acquire. The number must be greater than or equal to the first channel. In Single-ended (16 channels) input mode, the number must be less than or equal to 16. In Differential (8 channels) input mode, the number must be less than or equal to 8.

### Input coupling

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

Check the hardware manual of the board for wiring configurations.

### Gain vector

Enter a vector of gain values with one entry for each channel in the range first channel to last channel. Allowable gain settings for the AIM16 are 1, 2, 4, and 8. If you enter a scalar for gain, this gain value is used for all channels.

### Sample time

Enter a base sample time or a multiple of the base sample time.

# Analogic AIM16 Analog Input (A/D)

---

## **Base address**

Enter the base address of the board in hexadecimal (such as 0x300) as determined by the hardware jumpers on the board.



**Purpose** AIM16 Digital Input block

**Library** xPC Target Library for Analog

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 8 to select the digital input lines to be read. 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 of the digital inputs enter

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

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

### Channel group

Choose Lower 8 bits to connect to channels 1 through 8 or Upper 8 bits to connect to channels 9 through 16.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### Base address

Enter the base address of the board in hexadecimal (such as 0x300) as determined by the hardware jumpers on the board.

# Analogic AIM16 Digital Output

---

**Purpose** AIM16 Digital Output

**Library** xPC Target Library for Analog

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 8 to select the digital input lines to be read. 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 of the digital inputs enter

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

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

### Channel group

Choose Lower 8 bits to connect to channels 1 through 8 or Upper 8 bits to connect to channels 9 through 16.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### Base address

Enter the base address of the board in hexadecimal (such as 0x300) as determined by the hardware jumpers on the board.

# BittWare

---

This chapter describes the BittWare boards supported by xPC Target (<http://www.bittware.com>). Note that you can no longer buy this board from BittWare. The MathWorks supplies and supports the drivers and documentation for the customers who already own this board.

“Running Models with BittWare Blocks” on page 14-2

Usage notes to help use the BittWare blocks in your model.

Bittware Audio-PMC+

Professional audio board with 8 channels of audio input and output.

“Boards and Blocks — Alphabetical List” on page 14-6

Description of block parameters for BittWare driver blocks

## Running Models with BittWare Blocks

This topic describes usage notes to help use the BittWare blocks in your model.

- “Model Notes” on page 14-2
- “Frame Size, Sample Rate, and Sampletime Parameter Notes” on page 14-4

### Model Notes

This topic describes notes on working with models that use the BittWare blocks.

- “Model Execution Timing” on page 14-2
- “Example Models” on page 14-3
- “Model Execution Limitations” on page 14-4

### Model Execution Timing

To run the model, the xPC Target model is executed when each frame completes on the board. Set the simulation parameters to use the interrupt from the Audio-PMC+ rather than the timer interrupt when running the model. First, you need to determine the interrupt vector number to which the board is set. This is determined by the BIOS in the target machine during boot.

- 1 From the MATLAB Command Window, type

```
getxpcpci
```

This command lists board information for all installed PCI devices that xPC Target knows about.

- 2 Find the IRQ specified for this board. This is the interrupt source number you need to specify in the **xPC target code generation options** field in step of the following procedure.

Set the interrupt vector number:

- 1 From the MATLAB Command Window, type the name of your Simulink model.

The Simulink model appears

**2** From the **Simulation** menu, click **Configuration Parameters**.

The Configuration Parameter dialog box is displayed for the model.

**3** Click the **xPC Target options** node.

**4** Ensure that the **Execution mode** field is set to Real-Time.

**5** Click the **Real-time interrupt source** list.

**6** Select the interrupt number to which the board is set (from step in the previous procedure).

**7** Click the **I/O board generating the interrupt** list and select AudioPMC+ from the list.

**8** Click **OK** and save the model.

Failure to set the interrupt vector as described, which results in your using the xPC Target timer, will not stop the model from running. However, you will occasionally see corrupt data since the Audio-PMC+ clock and the xPC Target timer drift relative to each other and will sometimes overlap.

## Example Models

Two example models are included with this version in the `xpcdemos` directory. Correct operation of either of these requires that the interrupt source be set correctly in the simulation parameters dialog.

`xpc8audiochannels.mdl` shows the use of all 8 input and 8 output channels at the same time. A block from the Signal Processing Blockset is used to convert from frame to sample based signals to drive the xPC Target scopes.

`xpcaudiospectrum.mdl` uses the xPC Target Spectrum Scope subsystem from the `xpcdsppectrum` sample model with the Audio-PMC+ as the source. This model only works correctly with Frame Size set to 1 because of the way that the spectrum scope displays the results.

## Model Execution Limitations

Model execution is entered each time a frame completes on the Audio-PMC+ board. The frame rate is the fastest clock available. You cannot obtain a minor step execution that runs more frequently than frame completions. Any attempt to do a multirate model must use the frame completion time as the fastest rate. The audio input and audio output blocks must be executed at the fastest rate in the model.

Input and output data from these blocks is in the form of a frame of data as used by the Signal Processing Blockset. If the frame size is 1, then the blocks revert to sample based signals. The blocks in the Signal Processing Blockset expect to see signals that are frames.

## Frame Size, Sample Rate, and Sampletime Parameter Notes

Experimentally, with all 8 input and all 8 output channels in use, the Audio-PMC+ can run with sample rates below the following frequencies as a function of Frame Size:

Frame Size	Maximum Sample Rate
1	15 KHz
2	30 KHz
3	40 KHz
4	45 KHz
8	70 KHz
16	80 KHz
32	80 KHz
64	100 KHz
128	100 KHz
256	100 KHz

If fewer than all 8 inputs and outputs are in use, then the maximum sample rate increases.

<b>Frame Size</b>	<b>1 Channel</b>	<b>2 Channels</b>	<b>4 Channels</b>	<b>6 Channels</b>	<b>8 Channels</b>
1	30 KHz	30 KHz	25 KHz	18 KHz	15 KHz
2	60 KHz	60 KHz	45 KHz	33 KHz	30 KHz

These above rates were obtained from testing at The MathWorks. As with all hardware benchmarks, results might vary, depending on a number of hardware conditions. For example, the following hardware elements, among others, might effect your throughput:

- The PCI bus interface chip set on the mother board
- Main memory speed
- General mother board architecture

## **Boards and Blocks – Alphabetical List**



**Board** BittWare Audio-PMC+

**Library** xPC Target Library for Bittware

**General Description** The Audio-PMC+ board is an audio board with 8 I/O channels. xPC Target supports this board with these driver blocks:

- Bittware Audio-PMC+ Analog Input
- Bittware Audio-PMC+ Analog Output

Block parameters for the input and output blocks appear the same.

<b>Board Characteristics</b>	Board name	Audio-PMC+
	Manufacturer	BittWare
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	No
	Multiple board support	No

- Features**
- Input and output are in the form of the Signal Processing Blockset frame signal type when the frame is larger than a single sample.
  - The analog input block outputs a double or 32 bit integer frame of data.
  - The Audio-PMC+ board performs acquisition simultaneously for all the selected channels in a model.
  - The analog output block is dynamically typed. The integer or double data type is taken from the connection instead of a block parameter.

## Hardware Installation Notes

The Audio-PMC+ daughter board might come from BittWare with jumpers installed to boot from an onboard ROM. You must remove these jumpers, as follows:

- 1** With the Audio-PMC+ off the carrier board, look for 6 pairs of pins along one edge. For proper operation with xPC Target, none of the pairs should be jumpered.
- 2** If there are any jumpers present, remove and then reinstall the jumpers with only one pin connected. Leave the other side of the jumper hanging. Leaving the jumpers like this will provide you with the jumpers if you want to use the board in ROM bootable mode at some future time.

**Purpose** Audio-PMC+ Analog Input block

**Library** xPC Target Library for Bittware

## **Block Parameters**

### **Channel Vector**

This is a vector of channels. Specifies the input channels that the block works on. For example, to use the first, third, and fifth analog input channels, enter

[1,3,5]

### **Output Format**

Specify the format that the data takes. The choices are

- A normalized double in the range from -1.0 to 1.0
- An unscaled integer

The unscaled integer format is usable with blocks that can accept fixed point data. This format contains the data in the lower 24 bits of a 32 bit integer with sign extension in the upper 8 bits.

This parameter is only settable for the analog input block. The analog output block determines the data type from the connection and sets itself appropriately.

### **Frame Size, Sample Rate, and Sampletime**

These three parameters are not independent but are related by:

$$FrameSize = Sampletime \times SampleRate$$

After you specify two of the parameters, the equation determines the third parameter.

For example, if you set **Frame Size** to 32 and **Sample Rate** to 40000, specify **Sampletime** as -1. It is computed internally to 0.0008 seconds. This example model will execute every 0.0008 seconds, which is every 32 samples at 40 KHz.

# Bittware Audio-PMC+ Analog Input

---

Conversely, you can also specify that you want to execute every 1.0 ms with a **Frame Size** of 64 samples. Specify **Sample Rate** as -1. It is computed to be 64000 Hz.

The implementation of **Sample Rate** on the Audio-PMC+ might cause the rate on your board to be inexact. Audio-PMC+ derives the sample rate using the PWM0 output from the first 21065L SHARC DSP chip. The base clock that effects this output is the 60 MHz CPU clock for sample rates between 100 KHz and 20 KHz and a prescaled 30 MHz CPU clock (half the 60 MHz CPU clock) for sample rates between 8 KHz and 20 KHz.

The sample rate is implemented internally by specifying the total number of counts per sample interval. For example, the number of counts for a sample rate of 44.1 KHz is calculated as

$$60 \times 10^6 / 44.1 \times 10^3 = 1360.544217$$

This value is rounded down to 1360. To calculate the actual clock sample rate, divide 60 MHz by the number of counts

$$60 \times 10^6 / 1360 = 44117.647$$

The result indicates that the sample rate is off by 0.04%. This calculation assumes that the CPU clock is exactly 60 MHz.

Since you can build a model with just input or just output, the three rate parameters are included in both block parameter dialogs. You must set them to equal values if you have a model that has both. If they are not equal, you will get an error when you try to build the model.

The practical limit on **Sample Rate** is

$$8000 \leq \text{SampleRate} \leq 100000$$

Below 8 KHz, the output digital filter does not seem to work, although the converter continues to work there.

**Frame Size** values can be between 1 and 256. If the specified or computed frame size exceeds 256, the driver will return an error during initialization.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

---

**Note** The use of multiple boards, at the same time and in the same model, is unsupported.

---

**See Also**

“Running Models with BittWare Blocks” on page 14-2

# Bittware Audio-PMC+ Analog Output

---

**Purpose** Audio-PMC+ Analog Output block

**Library** xPC Target Library for Bittware

## **Block Parameters**

### **Channel Vector**

This is a vector of channels. Specifies the output channels that the block works on. For example, to use the first, third, and fifth analog output channels, enter

[1,3,5]

### **Frame Size, Sample Rate, and Sampletime**

These three parameters are not independent but are related by:

$$\textit{FrameSize} = \textit{Sampletime} \times \textit{SampleRate}$$

After you specify two of the parameters, the equation determines the third parameter.

For example, if you set **Frame Size** to 32 and **Sample Rate** to 40000, specify **Sampletime** as -1. It is computed internally to 0.0008 seconds. This example model will execute every 0.0008 seconds, which is every 32 samples at 40 KHz.

Conversely, you can also specify that you want to execute every 1.0 ms with a **Frame Size** of 64 samples. Specify **Sample Rate** as -1. It is computed to be 64000 Hz.

The implementation of **Sample Rate** on the Audio-PMC+ might cause the rate on your board to be inexact. Audio-PMC+ derives the sample rate using the PWM0 output from the first 21065L SHARC DSP chip. The base clock that effects this output is the 60 MHz CPU clock for sample rates between 100 KHz and 20 KHz and a prescaled 30 MHz CPU clock (half the 60 MHz CPU clock) for sample rates between 8 KHz and 20 KHz.

# Bittware Audio-PMC+ Analog Output

---

The sample rate is implemented internally by specifying the total number of counts per sample interval. For example, the number of counts for a sample rate of 44.1 KHz is calculated as

$$60 \times 10^6 / 44.1 \times 10^3 = 1360.544217$$

This value is rounded down to 1360. To calculate the actual clock sample rate, divide 60 MHz by the number of counts

$$60 \times 10^6 / 1360 = 44117.647$$

The result indicates that the sample rate is off by 0.04%. This calculation assumes that the CPU clock is exactly 60 MHz.

Since you can build a model with just input or just output, the three rate parameters are included in both block parameter dialogs. You must set them to equal values if you have a model that has both. If they are not equal, you will get an error when you try to build the model.

The practical limit on **Sample Rate** is

$$8000 \leq \text{SampleRate} \leq 100000$$

Below 8 KHz, the output digital filter does not seem to work, although the converter continues to work there.

**Frame Size** values can be between 1 and 256. If the specified or computed frame size exceeds 256, the driver will return an error during initialization.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Bittware Audio-PMC+ Analog Output

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

---

**Note** The use of multiple boards, at the same time and in the same model, is unsupported.

---

### See Also

“Running Models with BittWare Blocks” on page 14-2



# BVM

---

This chapter describes the BVM I/O board supported by xPC Target (<http://www.bvmltd.co.uk>).

BVM PMCDIO64	A 64-bit digital I/O board with two ports. Each port can be configured to be either 32 independent 1-bit channels or a single 32-bit wide integer channel.
--------------	--

## **Boards and Blocks**

**Board** BVM PMCDIO64

**General Description**

The PMCDIO64 is a 64-bit digital I/O board. The hardware provides 64 bits, which the driver splits into two ports of 32 bits each. Each port can be configured to be either input or output. In addition, each port can be configured to be either 32 independent 1-bit channels or a single 32-bit wide integer channel.

xPC Target supports this board with these driver blocks:

- BVM PMCDIO64 Digital Input
- BVM PMCDIO64 Digital Output

**Board Characteristics**

Board name	PMCDIO64
Manufacturer	BVM
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Yes
Multiple board support	Yes

# BVM PMCDIO64 Digital Input

---

**Purpose** PMCDIO64 Digital Input block

**Library** xPC Target Library for BVM

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

## **Block Parameters**

### **Format**

From the list, select either 32 One bit channels or Single 32 bit port.

If the format is 32 One bit channels, then the channel vector specifies the configuration of the block. Each output is a double with value of 0 or 1. The value is 0 when the hardware input voltage is low.

If the format is Single 32 bit port, then the output from the block is a 32-bit integer where all 32 bits on the hardware feed into the single output. The channel vector is not used in this mode and is unavailable on the Block Parameters dialog box. The least significant bit is the lowest numbered bit in the hardware manual.

### **Channel vector**

This is a vector of channels. This parameter is only used when the format is 32 One bit channels. Channels are numbered from 1 to 32 even though the hardware manual labels them 0 to 31.

### **Port**

Each half of the 64 bits is a separate port. Port 1 is the lower 32 bits and Port 2 is the upper 32 bits. You can use one port for input and the other port for output.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# BVM PMCDIO64 Digital Output

---

**Purpose** PMCDIO64 Digital Output block

**Library** xPC Target Library for BVM

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Format**

From the list, choose either 32 One bit channels or Single 32 bit port.

If the format is 32 One bit channels, then the channel vector specifies the configuration of the block. Each input is a double. The hardware output is set to low voltage if the input is < 0.5 and high voltage if the input is ≥ 0.5.

If the format is Single 32 bit port, then the input to the block is a 32-bit integer where all 32 bits on the hardware are controlled by the single input. The channel vector is not used in this mode and is unavailable on the Block Parameters dialog box. The least significant bit is the lowest numbered bit in the hardware manual.

### **Channel vector**

This is a vector of channels and is only used when the format is 32 One bit channels. Channels are numbered from 1 to 32 even though the hardware manual labels them 0 to 31. Hardware I/O signal numbers IO32 to IO63 are acquired by choosing Port 2 and channels 1 to 32.

### **Reset action vector**

If you chose 32 One bit channels, enter a vector of 1's and 0's that is the same length as the channel vector. A value of 1 indicates that the channel is reset to the value in the initial value

vector when the model is stopped. A value of 0 indicates that the output remains at the last value written when the model is stopped. If you enter a scalar value, that value is used for all channels.

If you chose Single 32 bit port, enter a 1 or a 0 to determine what happens when the model is stopped. If you enter 1, all 32 bits of the output are reset to the value given by the initial value vector. If you enter 0, the output remains at the last value written when the model is stopped.

### **Initial value vector**

If you chose 32 One bit channels, this vector determines both the initial value of the outputs at xPC boot time and the values when model execution is stopped. A value of 1 for a given channel sets the output for that channel to 1 while any other value sets the output to 0.

If you chose Single 32 bit port, enter the scalar value to write to the output port. The value can be a hexadecimal or a decimal. If it is a hexadecimal, then use C syntax. For example, 0xaaaaaaaa in hexadecimal would be the equivalent of 2863311530 in decimal.

### **Port**

Each half of the 64 bits is a separate port. Port 1 is the lower 32 bits and Port 2 is the upper 32 bits. You can use one port for input and the other port for output. In a given model, each port can only be set to one direction.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## BVM PMCDIO64 Digital Output

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# Commtech

---

xPC Target uses the Commtech Fastcom:422/2-PCI (<http://www.comtech-fastcom.com>) board drivers as part of its solution for serial communications. These blocks are not described in this chapter. Instead, see Chapter 2, “Serial Communications Support” for details.



# Condor

---

xPC Target uses Condor (<http://www.condoreng.com>) board drivers as part of its solution for the ARINC 429 and MIL-STD-1553 buses. These blocks are not described in this chapter. Instead, see Chapter 7, “ARINC 429 Support” and Chapter 8, “MIL-STD-1553 Support” for details.



# Contec

---

This chapter describes the Contec I/O boards supported by xPC Target (<http://www.contec.com>).

Contec AD12-16(PCI)	I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12 bit), 4 digital input lines, and 4 digital output lines.
Contec AD12-16(PCI)E	I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12 bit), one analog output channel (12 bit), 4 digital input lines, 4 digital output lines, and an i8254-compatible counter.
Contec AD12-16U(PCI)E	I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12 bit), one analog output channel (12 bit), 4 digital input lines, 4 digital output lines, and an i8254-compatible counter.
Contec AD12-64(PCI)	I/O board with 64 single-ended or 32 differential analog input (A/D) channels (12 bit), 4 digital input lines, and 4 digital output lines.
Contec AD16-16(PCI)E	I/O board with 16 single-ended or 8 differential analog input (A/D) channels (16 bit), one analog output channel (16 bit), 4 digital input lines, 4 digital output lines, and an i8254-compatible counter.

Contec ADI12-16(PCI)	I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12 bit), 4 digital input lines, and 4 digital output lines.
Contec CNT24-4D(PCI)	24-bit differential up/down counter board with four channels.
Contec CNT32-8M(PCI)	32-bit high-speed up/down counter board with eight channels.
Contec DA12-16(PCI)	I/O board with 16 analog output (D/A) channels (12 bit).
Contec DA12-4(PCI)	I/O board with 4 analog output (D/A) channels (12 bit).
Contec PI-64L(PCI)H	Digital input board with 64 digital input channels.
Contec PIO-32/32F(PCI)	Optoisolated high-speed I/O board with 32 digital input channels and 32 output channels.
Contec PIO-32/32L(PCI)H	I/O board with 32 digital input channels and 32 output channels.
Contec PIO-32/32T(PCI)	I/O board with 32 digital input lines and 32 digital output lines.
Contec PO-64L(PCI)H	Digital output board with 64 digital output channels.
“Boards and Blocks — Alphabetical List” on page 18-3	Description of block parameters for Contec driver blocks.

## **Boards and Blocks – Alphabetical List**

# Contec AD12-16(PCI)

---

## Board

Contec AD12-16(PCI)

## General Description

The Contec AD12-16(PCI) is an I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12 bit), 4 digital input lines, and 4 digital output lines.

xPC Target supports this board with these driver blocks:

- Contec AD12-16(PCI) Analog Input (A/D)
- Contec AD12-16(PCI) Digital Input
- Contec AD12-16(PCI) Digital Output

xPC Target does not support the Counter/Timer functionality of the board.

## Board Characteristics

Board name	AD12-16(PCI)
Manufacturer	Contec
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes



# Contec AD12-16(PCI) Analog Input (A/D)

**Purpose** AD12-16(PCI) Analog Input (A/D) block

**Library** xPC Target Library for Contec

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

**Block Parameters** **Channel vector**  
Enter a vector of numbers to specify the input channels. For example to use the first three analog input (A/D) channels, enter

[1, 2, 3]

The channel numbers can occur in any order. Number them beginning with 1 even if the board manufacturer numbers them beginning with 0.

The maximum allowable channel number for this board is 8 (double-ended) or 16 (single-ended). If the highest channel number you specify is n, the hardware converts all the channels between 1 and n, whether or not they occur in your channel vector. It is most efficient to specify a contiguous range of channels. (Permuting the order of such a range has no impact on efficiency however.)

### **Range vector**

This board allows the range of each channel to be selected independently. Enter a scalar, in which case the same range will be used for all channels, or a vector the same length as the channel vector.

The range vector entries must be range codes selected from the following table:

## Contec AD12-16(PCI) Analog Input (A/D)

---

Input Range (V)	Range Code	Input Range (V)	Range Code
-10 to +10	-10	0 to 10	10
-5 to +5	-5	0 to 5	5
-2.5 to 2.5	-2	0 to 2.5	2
-1.25 to 1.25	-1	0 to 1.25	1

### Polarity

Choose single-ended or double-ended. This setting applies to all channels.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The Device ID of this board is 8153.

# Contec AD12-16(PCI) Digital Input

**Purpose** AD12-16(PCI) Digital Input block

**Library** xPC Target Library for Contec

## Scaling Input to Output

**Hardware  
Input**

**Block Output Data Type**

**Scaling**

TTL

Double

TTL low = 0.0

TTL high = 1.0

## Block Parameters

### Channel vector

Enter a vector of numbers to specify the input channels. For example to use the first and third digital input channels enter

[1, 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Contec AD12-16(PCI) Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The Device ID of this board is 8153.

# Contec AD12-16(PCI) Digital Output

**Purpose** AD12-16(PCI) Digital Output block

**Library** xPC Target Library for Contec

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 TTL high

**Block Parameters**

**Channel vector**  
Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels enter

[1, 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4.

**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.

# Contec AD12-16(PCI) Digital Output

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The Device ID of this board is 8153.

## Board

Contec AD12-16(PCI)E

## General Description

The Contec AD12-16(PCI)E is an I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12 bit), one analog output channel (12 bit), 4 digital input lines, 4 digital output lines, and an i8254-compatible counter.

xPC Target supports this board with these driver blocks:

- Contec AD12-16(PCI)E Analog Input (A/D)
- Contec AD12-16(PCI)E Analog Output (D/A)

xPC Target does not support the digital I/O or Counter/Timer functionality of the board.

## Board Characteristics

Board name	AD12-16(PCI)E
Manufacturer	Contec
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Contec AD12-16(PCI)E Analog Input (A/D)

---

**Purpose** AD12-16(PCI)E Analog Input block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first three analog input (A/D) channels, enter

[1, 2, 3]

The channel numbers can occur in any order. Number them beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for this board is 8 (double-ended) or 16 (single-ended).

### **Gain vector**

To specify the gain, enter 1, 2, 4, or 8 for each of the channels in the channel vector. The gain vector must be the same length as the channel vector. If you enter a scalar, the value is applied to all channels.

The gain is applied to the signal prior to sampling the voltage. After the signal voltage is sampled, the result is divided by the gain to obtain the block output signal value. To avoid clipping, make sure that the amplified gain falls within the range you selected.

### **Range**

From the list, select Bipolar -10V to +10V or Unipolar 0V to +10V. This range applies to all channels.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.



# Contec AD12-16(PCI)E Analog Input (A/D)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of this board is 8113.

# Contec AD12-16(PCI)E Analog Output (D/A)

---

**Purpose** AD12-16(PCI)E Analog Output block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Range**

From the list, select -5V to +5V, -10V to +10V, or 0 to +10V as the output range for the analog output. This range must correspond to the output range determined by the jumpers on the board.

### **Reset**

This check box controls the behavior of the output channel at model termination. If the check box is selected, the output channel is reset to the value specified as the initial value. If the check box is not selected, the output channel remains at the most recent value attained while the model was running.

### **Initial value**

Enter the initial voltage value for the output channel. The output channel is set to this value between the time the model is downloaded and the time the model is started. Also, if the reset check box is selected, the output channel is reset to the initial value when the model is stopped.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Contec AD12-16(PCI)E Analog Output (D/A)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of this board is 8113.

# Contec AD12-16U(PCI)E

---

## Board

Contec AD12-16U(PCI)E

## General Description

The Contec AD12-16U(PCI)E is an I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12 bit), one analog output channel (12 bit), 4 digital input lines, 4 digital output lines, and an i8254-compatible counter.

xPC Target supports this board with these driver blocks:

- Contec AD12-16U(PCI)E Analog Input (A/D)
- Contec AD12-16U(PCI)E Analog Output (D/A)

xPC Target does not support the digital I/O or Counter/Timer functionality of the board.

## Board Characteristics

Board name	AD12-16U(PCI)E
Manufacturer	Contec
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Contec AD12-16U(PCI)E Analog Input (A/D)

**Purpose** AD12-16U(PCI)E Analog Input block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first three analog input (A/D) channels, enter

[1, 2, 3]

The channel numbers can occur in any order. Number them beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for this board is 8 (double-ended) or 16 (single-ended).

### **Range**

From the list, select Bipolar -2.5V to +2.5V, Bipolar -5V to +5V, Unipolar 0V to +5V, or Unipolar 0V to +10V. This range applies to all channels.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Contec AD12-16U(PCI)E Analog Input (A/D)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of this board is 8103.

# Contec AD12-16U(PCI)E Analog Output (D/A)

**Purpose** AD12-16U(PCI)E Analog Output block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Range**

From the list, select -5V to +5V, -10V to +10V, or 0 to +10V as the output range for the analog output. This range must correspond to the output range determined by the jumpers on the board.

### **Reset**

This check box controls the behavior of the output channel at model termination. If the check box is selected, the output channel is reset to the value specified as the initial value. If the check box is not selected, the output channel remains at the most recent value attained while the model was running.

### **Initial value**

Enter the initial voltage value for the output channel. The output channel is set to this value between the time the model is downloaded and the time the model is started. Also, if the reset check box is selected, the output channel is reset to the initial value when the model is stopped.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Contec AD12-16U(PCI)E Analog Output (D/A)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of this board is 8103.



**Purpose** Contec AD12-64(PCI) block

**General Block Description** The Contec AD12-64(PCI) is an I/O board with 64 single-ended or 32 differential analog input (A/D) channels (12 bit), 4 digital input lines, and 4 digital output lines.

xPC Target supports this board with these driver blocks:

- Contec AD12-64(PCI) Analog Input (A/D)
- Contec AD12-64(PCI) Digital Input
- Contec AD12-64(PCI) Digital Output

xPC Target does not support the Counter/Timer functionality of this board.

<b>Board Characteristics</b>	Board name	AD12-64(PCI)
	Manufacturer	Contec
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# Contec AD12-64(PCI) Analog Input (A/D)

---

**Purpose** AD12-64(PCI) Analog Input block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first three analog input (A/D) channels, enter

[1, 2, 3]

The channel numbers can occur in any order. Number them beginning with 1 even if the board manufacturer numbers them beginning with 0.

The maximum allowable channel number for this board is 32 (double-ended) or 64 (single-ended). If the highest channel number you specify is  $n$ , the hardware will convert all the channels between 1 and  $n$ , whether or not they occur in your channel vector. It is most efficient to specify a contiguous range of channels. (Permuting the order of such a range has no impact on efficiency however.)

# Contec AD12-64(PCI) Analog Input (A/D)

## Range vector

This board allows the range of each channel to be selected independently. If you enter a scalar, the same range is used for all channels. If you enter a vector, it must be the same length as the channel vector.

The range vector entries must be range codes selected from the following table:

Input Range (V)	Range Code	Input Range (V)	Range Code
-10 to +10	-10	0 to 10	10
-5 to +5	-5	0 to 5	5
-2.5 to 2.5	-2	0 to 2.5	2
-1.25 to 1.25	-1	0 to 1.25	1

## Polarity

Choose single-ended or double-ended. This setting applies to all channels.

## Sample time

Enter the base sample time or a multiple of the base sample time.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Contec AD12-64(PCI) Analog Input (A/D)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The Device ID of this board is 8143.

# Contec AD12-64(PCI) Digital Input

**Purpose** AD12-64(PCI) Digital Input block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels enter

[1, 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The Device ID of this board is 8143.

# Contec AD12-64(PCI) Digital Output

---

**Purpose** AD12-64(PCI) Digital Output block

**Library** xPC Target Library for Contec

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 TTL high

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels enter

[1, 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4.

### **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 (0 or 1) 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.

# Contec AD12-64(PCI) Digital Output

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The Device ID of this board is 8143.

# Contec AD16-16(PCI)E

---

## Board

Contec AD16-16(PCI)E

## General Description

The Contec AD16-16(PCI)E is an I/O board with 16 single-ended or 8 differential analog input (A/D) channels (16 bit), one analog output channel (16 bit), 4 digital input lines, 4 digital output lines, and an i8254-compatible counter.

xPC Target supports this board with these driver blocks:

- Contec AD16-16(PCI)E Analog Input (A/D)
- Contec AD16-16(PCI)E Analog Output (D/A)

xPC Target does not support the digital I/O or Counter/Timer functionality of the board.

## Board Characteristics

Board name	AD16-16(PCI)E
Manufacturer	Contec
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes



# Contec AD16-16(PCI)E Analog Input (A/D)

**Purpose** AD16-16(PCI)E Analog Input block

**Library** xPC Target Library for Contec

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

## Block Parameters

### Channel vector

Enter a vector of numbers to specify the input channels. For example, to use the first three analog input (A/D) channels, enter

[1, 2, 3]

The channel numbers can occur in any order. Number them beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for this board is 8 (double-ended) or 16 (single-ended).

### Range

From the list, select Bipolar -10V to +10V, Bipolar -5V to +5V, Unipolar 0V to +5V, or Unipolar 0V to +10V. This range applies to all channels.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Contec AD16-16(PCI)E Analog Input (A/D)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of this board is 8123.

# Contec AD16-16(PCI)E Analog Output (D/A)

**Purpose** AD16-16(PCI)E Analog Output block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Range**

From the list, select -5V to +5V, -10V to +10V, or 0 to +10V as the output range for the analog output. This range must correspond to the output range determined by the jumpers on the board.

### **Reset**

This check box controls the behavior of the output channel at model termination. If the check box is selected, the output channel is reset to the value specified as the initial value. If the check box is not selected, the output channel remains at the most recent value attained while the model was running.

### **Initial value**

Enter the initial voltage value for the output channel. The output channel is set to this value between the time the model is downloaded and the time the model is started. Also, if the reset check box is selected, the output channel is reset to the initial value when the model is stopped.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Contec AD16-16(PCI)E Analog Output (D/A)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of this board is 8123.

**Board** Contec ADI12-16(PCI)

**General Description** The Contec ADI12-16(PCI) is an I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12 bit), 4 digital input lines, and 4 digital output lines.

xPC Target supports this board with these driver block:

- Contec ADI12-16(PCI) Analog Input (A/D)

<b>Board Characteristics</b>	Board name	ADI12-16(PCI)
	Manufacturer	Contec
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# Contec ADI12-16(PCI) Analog Input (A/D)

---

**Purpose** ADI12-16(PCI) Analog Input block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first three analog input (A/D) channels, enter

[1, 2, 3]

The channel numbers can occur in any order. Number them beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for this board is 8 (double-ended) or 16 (single-ended).

### **Gain vector**

To specify the gain, enter 1, 2, 4, or 8 for each of the channels in the channel vector. The gain vector must be the same length as the channel vector. If you enter a scalar, the value is applied to all channels.

The gain is applied to the signal prior to sampling the voltage. After the signal voltage is sampled, the result is divided by the gain to obtain the block output signal value. To avoid clipping, make sure that the amplified gain falls within the range you selected.

### **Range**

From the list, select Bipolar -10V to +10V, Unipolar 0V to +10V, or Unipolar 4mA to 20mA. Choose a range that is consistent with the jumper settings on the board. This range applies to all channels.

# Contec ADI12-16(PCI) Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of this board is 8133.

# Contec CNT24-4D(PCI)

---

## Board

Contec CNT24-4D(PCI)

## General Description

The Contec CNT24-4D(PCI) is a 24-bit differential up/down counter board with four 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.

xPC Target supports this board with one driver block:

- Contec CNT24-4D(PCI) Incremental Encoder

xPC Target does not support the timer or interrupt functionality of this board.

## Board Characteristics

Board name	CNT24-4D(PCI)
Manufacturer	Contec
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes



# Contec CNT24-4D(PCI) Incremental Encoder

---

**Purpose** CNT24-4D(PCI) Incremental Encoder block

**Library** xPC Target Library for Contec

**Block Parameters** Note that you can have only one driver block instance for each physical board. If you need to use multiple channels, select and enable more than channel on the same block.

## Channel

From the list, select 1, 2, 3, or 4. This parameter specifies the channel to which the subsequent parameters refer. The other parameters in this block apply to this channel. The **Enable channel** check box number changes to match the selected channel.

## Enable channel

Select this check box to enable the currently selected channel. Click **OK** after you select this check box to add an output port for the channel on the driver instance of your model. This check box also enables you to set a number of operation parameters for the block, ranging from **Input type** to **Initial count**. Whatever operation parameters are in place when you click **OK** are preserved for the channel until the next time you change them.

The following are some additional behavior notes for this check box:

- If you do not select this check box for a channel, an output port for that channel is not added to the block. You also cannot change the operation parameters for the channel.
- If you select this check box and save the operation parameters for that channel, then later deselect this check box, the block preserves the operation parameters for the channel. The output port is removed from the block.

## Input type

From the list, choose either `Line receiver` or `TTL-level input`. This parameter specifies the input type for the current channel.

# Contec CNT24-4D(PCI) Incremental Encoder

---

## **Mode**

From the list, select the counter operation mode for the current channel. There are a number of modes, based on 1-phase or 2-phase pulse inputs. See the Contec CNT24-4D(PCI) user's guide documentation for descriptions of these modes.

Note that of this list of modes, the Multiply by 4 modes are synonyms for quadrature encoding.

## **Direction**

From the list, select either Clockwise rotation counts down or Clockwise rotation counts up as the counter direction of the current channel.

## **Phase Z logic**

From the list, select either Active high or Active low for the current channel. This parameter specifies the state of the phase Z input (reference position signal). If the **phase Z mode** parameter has a value of Disable phase Z input, **Phase Z logic** has no effect.

## **Phase Z mode**

From the list, select either Disable phase Z input, Enable next phase Z input only once, or Enable every phase Z input. This parameter specifies the operation mode of the phase Z input for the current channel.

## **Digital filter**

From the list, select the characteristics of the digital input filter you want to apply to the current channel's input signal. There are a number of sampling cycles to choose from, ranging from 0.1 microseconds (1 MHz) to 1056.1 microseconds (94 Hz).

## **Initial count**

Enter a number from 0 to 16777215 (FFFFFF hex, the largest 24-bit number). This parameter specifies the initial value of the counter for the current channel.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Contec CNT24-4D(PCI) Incremental Encoder

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The Device ID of this board is 8163.

# Contec CNT32-8M(PCI)

---

**Board** Contec CNT32-8M(PCI)

**General Description** The Contec CNT32-8M(PCI) is a 32-bit high-speed up/down counter board with eight 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. xPC Target supports this board with this driver block:

- Contec CNT32-8M(PCI) Incremental Encoder

<b>Board Characteristics</b>	Board name	CNT32-8M(PCI)
	Manufacturer	Contec
	Bus type	PCI
	Multiple block instance support	Per model — Yes Per channel — No. To support multiple channels, use multiple blocks with each block configured to a different channel.
	Multiple board support	Yes

# Contec CNT32-8M(PCI) Incremental Encoder

**Purpose** CNT32-8M(PCI) Incremental Encoder block

**Library** xPC Target Library for Contec

**Note** Each channel of this block has an external input and output signal. You can configure these channels for hardware events (such as starting the encoder on the rising or falling edge of an input signal). You can also configure them for general purpose, where the signals behave like digital I/O signals. The block can have the following ports:

- Output port labeled with the channel number
- Output port labeled `din` , if you configure the block for digital input
- Input port labeled `dout` , if you configure the block for digital output

**Scaling** Encoder (Input to Output)

<b>Hardware Input</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
TTL or differential (pulse input)	Double	Counts

Digital Input (Input to Output)

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

Digital Output (Output to Input)

# Contec CNT32-8M(PCI) Incremental Encoder

---

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

## Block Parameters: Main Tab

The **Main** tab contains parameters that define the proper operation of the board.

### Channel

From the list, select 1 through 8. This parameter specifies the board channel to which the subsequent parameters refer. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The other parameters in this block apply to this channel. The channel you select becomes the label for the output port of the block.

### Operation mode

From the list, select one of the following operation modes:

- 2-phase Input Synchronous Clear Multiply by 1
- 2-phase Input Synchronous Clear Multiply by 2
- 2-phase Input Synchronous Clear Multiply by 4
- 2-phase Input Asynchronous Clear Multiply by 1
- 2-phase Input Asynchronous Clear Multiply by 2
- 2-phase Input Asynchronous Clear Multiply by 4
- Single-phase Input Asynchronous Clear Multiply by 1
- Single-phase Input with Gate Control Attached Asynchronous Clear Multiply 1
- Single-phase Input with Gate Control Attached Asynchronous Clear Multiply 2

# Contec CNT32-8M(PCI) Incremental Encoder

---

See the Contec CNT32-8M(PCI) 8-Ch 32-Bit Up/Down High-Speed Counter Board for PCI documentation for definitions of these modes.

## Digital filter

From the list, select the desired digital filter time, in microseconds. For no digital filter time, select No filtering.

## Pulse input

From the list, select one of the following pulse inputs:

- Differential line receiver input
- TTL Level input

You can connect the CNT32-8M(PCI) board to either differential or TTL input signals. See the Contec CNT32-8M(PCI) 8-Ch 32-Bit Up/Down High-Speed Counter Board for PCI documentation for connection details.

## Direction

From the list, select one of the following count directions:

- Up in the clockwise direction/Down in the counterclockwise direction
- Down in the clockwise direction/Up in the counterclockwise direction

The board can count either up or down in the clockwise direction.

## Initial value

Enter a number from 0 to the largest 32-bit number. This parameter specifies the initial value of the counter before the model begins.

## Load initial value

Select this check box to load the value from **Initial value** into the counter, overwriting the previous count value. If you do not select this check box, the driver does not load the initial value, preserving the previous count value.

# Contec CNT32-8M(PCI) Incremental Encoder

---

## Sample Time

Enter the base sample time or a multiple of the base sample time.

## PCI slot

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Block Parameters: Counter Control Tab

The **Counter Control** tab contains parameters that define

- How to start and stop counting
- How to use the external input signal
- Under what circumstances to reset the counter

## Start method

Directs counter to begin counting under one of the following conditions:

- Model Start — Begin counting when model starts.
- External Input Rising — Begin counting when external TTL input rises.
- External Input Falling — Begin counting when external TTL input falls.
- Disable — Do not begin counting.

## Stop method

Directs counter to stop counting under one of the following conditions:



# Contec CNT32-8M(PCI) Incremental Encoder

---

- **Model Terminate** — Stop counting when model stops.
- **External Input Rising** — Stop counting when external TTL input rises.
- **External Input Falling** — Stop counting when external TTL input falls.
- **Free Running** — Do not stop counting.

## **Zero-clear external input rising**

Select this check box to clear the counter upon a rising edge on the external input signal.

## **Zero-clear external input falling**

Select this check box to clear the counter upon a falling edge on the external input signal.

## **Zero-clear match 0**

Select this check box to clear the counter when the counter has the same value as that contained in the **Match 0 value** parameter.

## **Zero-clear match 1**

Select this check box to clear the counter when the counter has the same value as that contained in the **Match 1 value** parameter.

## **Preset external input rising**

Select this check box to preset the counter to the value contained in the **Preset value** parameter upon a rising edge on the external input signal.

## **Preset external input falling**

Select this check box to preset the counter to the value contained in the **Preset value** parameter upon a falling edge on the external input signal.

## **Preset match 0**

Select this check box to preset the counter to the value contained in the **Preset value** parameter when the counter value has the same value as that contained in the **Match 0 value** parameter.

# Contec CNT32-8M(PCI) Incremental Encoder

---

## **Preset match 1**

Select this check box to preset the counter to the value contained in the **Preset value** parameter when the counter value has the same value as that contained in the **Match 1 value** parameter.

## **Preset value**

Enter the value that you want the counter to reset when a preset condition occurs.

## **Match 0 value**

Enter the value that you want the counter value to be compared to for the **Zero-clear match 0** or **Preset match 0** conditions.

## **Match 1 value**

Enter the value that you want the counter value to be compared to for the **Zero-clear match 1** or **Preset match 1** conditions.

## **Block Parameters: Signals Tab**

The **Signals** tab contains parameters that

- Control the optional use of the external output signal
- Control the encoder index signal
- Enable the external signals for general purpose

## **Output signal match 0**

Select this check box to strobe the output signal when the counter value has the same value as the **Match 0 parameter**.

## **Output signal match 1**

Select this check box to strobe the output signal when the counter value has the same value as the **Match 1 parameter**.

## **Output signal abnormal input error**

Select this check box to strobe the output signal when the phase-A and phase-B signals change at the same time.

## **Output signal digital filter error**

Select this check box to strobe the output signal when a pulse is faster than the digital filter time setting.

# Contec CNT32-8M(PCI) Incremental Encoder

---

## Output signal disconnection alarm error

Select this check box to strobe the output signal when both the positive and negative differential inputs are high.

## One-shot duration

From the list, select the pulse width of the output signal for the one-shot duration, in microseconds or milliseconds.

## Index enable

From the list, select the following to enable or disable the phase-Z/CLR input signal:

- **Disable** — Disables phase-Z input (select this option if phase-Z does not exist).
- **Enable Once** — Enables only the next phase-Z input.
- **Enable All** — Enables all phase-Z inputs.

## Index logic

From the list, select one of the following to set the logic of the phase-Z input signal. Setting **Index enable** to **Enable Once** or **Enable All** activates this parameter.

- **Positive (Active High)**
- **Negative (Active Low)**

## Enable digital input

Select this check box to enable the external input signal to be used as a general purpose digital input. Even if you select an external input signal other than general-purpose input, the block can still read the input level.

## Enable digital output

Select this check box to enable the external output signal to be used as a general-purpose digital output. If you select an external output signal other than general purpose, the block cannot write the output level. Selecting this check box enables the **Digital output initial value** and **Digital output final value** parameters.

# Contec CNT32-8M(PCI) Incremental Encoder

---

## **Digital output initial value**

From the list, select how you want to set the initial value of the digital output when the model starts:

- None — Do not modify the output value.
- Set — Set the bit.
- Clear — Clear the bit.

## **Digital output final value**

From the list, select how you want to set the final value of the digital output when the model terminates:

- None — Do not modify the output value.
- Set — Set the bit.
- Clear — Clear the bit.

**Purpose** Contec DA12-16(PCI) block

**General Description** The Contec DA12-16(PCI) is an I/O board with 16 analog output (D/A) channels (12 bit).

xPC Target supports this board with this driver block:

- Contec DA12-16(PCI) Analog Output (D/A)

xPC Target does not support the timer, external trigger, or interrupt functionality of this board.

<b>Board Characteristics</b>	Board name	DA12-16(PCI)
	Manufacturer	Contec
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# Contec DA12-16(PCI) Analog Output (D/A)

---

**Purpose** DA12-16(PCI) Analog Output block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and second analog output (D/A) channels enter

[1, 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16.

### **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.

# Contec DA12-16(PCI) Analog Output (D/A)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The Device ID of this board is 8163.

# Contec DA12-4(PCI)

---

**Board** Contec DA12-4(PCI)

**General Description** The Contec DA12-4(PCI) is an I/O board with 4 analog output (D/A) channels (12 bit).

xPC Target supports this board with this driver block:

- Contec DA12-4(PCI) Analog Output (D/A)

xPC Target does not support the timer, external trigger, or interrupt functionality of this board.

<b>Board Characteristics</b>	Board name	DA12-4(PCI)
	Manufacturer	Contec
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes



# Contec DA12-4(PCI) Analog Output (D/A)

**Purpose** DA12-4(PCI) Analog Output block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and second analog output (D/A) channels enter

[1, 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4.

### **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.

# Contec DA12-4(PCI) Analog Output (D/A)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The Device ID of this board is 8183.

**Board** Contec PI-64L(PCI)H

**General Description**

The Contec PI-64L(PCI)H is a digital input board with 64 digital input channels. The supporting block can control up to 32 bits that you can configure as either a single 32-bit port (Single 32-bit Port mode) or up to 32 1-bit channels (32 1-bit Channels mode).

xPC Target supports this board with this driver block:

- Contec PI-64L(PCI)H Digital Input

**Board Characteristics**

Board name	PI-64L(PCI)H
Manufacturer	Contec
Bus type	PCI
Multiple block instance support	No
Multiple board support	Yes

# Contec PI-64L(PCI)H Digital Input

---

**Purpose** PI-64L(PCI)H Digital Input block

**Library** xPC Target Library for Contec

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double ( <b>Format</b> :32 1-bit Channels)	Double: TTL low = 0.0 TTL high = 1.0
	uint32 ( <b>Format</b> :Single 32-bit Port)	uint32: TTL low corresponding bit is clear TTL high corresponding bit is set

## Block Parameters

### Format

From the list, select either

- **32 1-bit Channels** — Indicates that the channel vector specifies the configuration of the block. Each output is a double with value of 0 or 1. The value is 0 when the hardware input voltage is low.
- **Single 32-bit Port** — Indicates that the output from the block is a 32-bit integer where all 32 bits on the board feed into the single output. The least significant bit is the lowest numbered bit in the Contec documentation.

If you choose this mode, use the bit packing and unpacking blocks (Digital I/O Bit-Packing and Digital I/O Bit-Unpacking) to construct appropriate data frames.

## Group

From the list, select either

- A (Bits 0-31) — Selects the 32-bit word from group A to address.
- B (Bits 32-63) — Selects the 32-bit word from group B to address.

## Channel vector

(32 1-bit Channels mode only) Enter a vector of numbers to specify the output channels. For example, to use the first and third digital input channels, enter

[1, 3]

This vector indexes into group A or B channels. For group A, channel vector [1, 3] specifies channels 0 and 2. For group B, channel vector [1, 3] specifies channels 32 and 34.

The channel numbers can occur in any order, but the numbers must be in the range 1 to 32.

## Digital Filter

This parameter specifies the digital filter time. From the list, select

- No filtering
- 0.25 usec
- 0.5 usec
- 1 usec
- 2 usec
- 4 usec
- 8 usec

See the Contec documentation for further details on the digital filter time.

# Contec PI-64L(PCI)H Digital Input

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** Contec PIO-32/32F(PCI)

**General Description** The Contec PIO-32/32F(PCI) is an optoisolated high-speed I/O board with 32 digital input channels and 32 output channels.

xPC Target supports this board with these driver blocks:

- Contec PIO-32/32T(PCI) Digital Input
- Contec PIO-32/32T(PCI) Digital Output

xPC Target does not support the timer, external trigger, or interrupt functionality of this board.

<b>Board Characteristics</b>	Board name	PIO-32/32F(PCI)
	Manufacturer	Contec
	Bus type	PCI
	Multiple block instance support	No
	Multiple board support	Yes

# Contec PIO-32/32F(PCI) Digital Input

---

**Purpose** PIO-32/32F(PCI) Digital Input block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **I/O format**

Select serial or parallel. If you select serial, the block is configured to accept up to 32 one-bit input channels. If you select parallel, the block is configured to accept a single 32-bit input channel and the channel vector parameter is unavailable.

### **Channel vector**

If you selected serial I/O format, enter a vector of numbers to specify the input channels for serial I/O format. For example, to use the first and third digital input channels, enter

[1, 3]

The channel numbers can occur in any order, but the numbers must be in the range 1 to 32.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.



# Contec PIO-32/32F(PCI) Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The device ID of this board is 8152.

# Contec PIO-32/32F(PCI) Digital Output

---

**Purpose** PIO-32/32F(PCI) Digital Output block

**Library** xPC Target Library for Contec

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **I/O format**

Select serial or parallel. If you select serial, the block is configured to accept up to 32 one-bit input channels for output. If you select parallel, the block is configured to accept a single 32-bit channel and the channel vector parameter is unavailable.

### **Channel vector**

For serial I/O format, enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1, 3]

The channel numbers can occur in any order, but the numbers must be in the range 1 to 32.

### **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 (0 or 1) 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 selected parallel I/O format, the values can be in the form [hex2dec('ffffff')].

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The device ID of this board is 8152.

# Contec PIO-32/32L(PCI)H

---

## Board

Contec PIO-32/32L(PCI)H

## General Description

The Contec PIO-32/32L(PCI)H is an I/O board with 32 digital input channels and 32 output channels. Each supporting block can control up to 32 bits that you can configure as either a single 32-bit port (Single 32-bit Port mode) or up to 32 1-bit channels (32 1-bit Channels mode).

xPC Target supports this board with these driver blocks:

- Contec PIO-32/32L(PCI)H Digital Input
- Contec PIO-32/32L(PCI)H Digital Output

xPC Target does not support the timer, external trigger, or interrupt functionality of this board.

## Board Characteristics

Board name	PIO-32/32L(PCI)H
Manufacturer	Contec
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

# Contec PIO-32/32L(PCI)H Digital Input

**Purpose** PIO-32/32L(PCI)H Digital Input block

**Library** xPC Target Library for Contec

## Scaling Input to Output

### Hardware Input

TTL

### Block Output Data Type

Double (**Format**:32 1-bit Channels)

uint32 (**Format**:Single 32-bit Port)

### Scaling

Double:  
TTL low = 0.0

TTL high = 1.0

uint32:  
TTL low corresponding bit is clear

TTL high corresponding bit is set

## Block Parameters

### Format

From the list, select either

- **32 1-bit Channels** — Indicates that the channel vector specifies the configuration of the block. Each output is a double with value of 0 or 1. The value is 0 when the hardware input voltage is low.
- **Single 32-bit Port** — Indicates that the output from the block is a 32-bit integer where all 32 bits on the board feed into the single output. The least significant bit is the lowest numbered bit in the Contec user manual.

If you choose this mode, use the bit packing and unpacking blocks (Digital I/O Bit-Packing and Digital I/O Bit-Unpacking) to construct appropriate data frames.

# Contec PIO-32/32L(PCI)H Digital Input

---

## **Channel vector**

(32 1-bit Channels mode only) Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1, 3]

The channel numbers can occur in any order, but the numbers must be in the range 1 to 32.

## **Digital Filter**

This parameter specifies the digital filter time. From the list, select

- No filtering
- 0.25 usec
- 0.5 usec
- 1 usec
- 2 usec
- 4 usec
- 8 usec

See the Contec User Guide for further details on the digital filter time.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Contec PIO-32/32L(PCI)H Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Contec PIO-32/32L(PCI)H Digital Output

---

**Purpose** PIO-32/32L(PCI)H Digital Output block

**Library** xPC Target Library for Contec

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double ( <b>Format</b> :32 1-bit Channels)  uint32 ( <b>Format</b> :Single 32-bit Port)	Double: < 0.5 = TTL low  >0.5 = TTL high  uint32: Bit clear = TTL low  Bit set = TTL high

## **Block Parameters**

### **Format**

From the list, select either

- **32 1-bit Channels** — Indicates that the channel vector specifies the configuration of the block. Each output is a double with value of 0 or 1. The value is 0 when the hardware input voltage is low.
- **Single 32-bit Port** — Indicates that the output from the block is a 32-bit integer where all 32 bits on the board feed into the single output. The least significant bit is the lowest numbered bit in the Contec user manual.

If you choose this mode, use the bit packing and unpacking blocks (Digital I/O Bit-Packing and Digital I/O Bit-Unpacking) to construct appropriate data frames.



## Channel vector

(32 1-bit Channels mode only) Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1, 3]

The channel numbers can occur in any order, but the numbers must be in the range 1 to 32.

## 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.

## Final action vector

The final action 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 -1, the block sets the channel to the value specified in the **Final value vector** value for that channel. If you specify a value of 0, the channel remains at the last value attained while the model was running.

## Final value vector

The final value vector contains the final value for each output channel. 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 the **Final action vector** is -1, the block sets the channel to this value on model termination.

## Sample time

Enter the base sample time or a multiple of the base sample time.

# Contec PIO-32/32L(PCI)H Digital Output

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** Contec PIO-32/32T(PCI)

**General Description** The Contec PIO-32/32T(PCI) is an I/O board with 32 digital input channels and 32 output channels.

xPC Target supports this board with these driver blocks:

- Contec PIO-32/32T(PCI) Digital Input
- Contec PIO-32/32T(PCI) Digital Output

xPC Target does not support the timer, external trigger, or interrupt functionality of this board.

<b>Board Characteristics</b>	Board name	PIO-32/32T(PCI)
	Manufacturer	Contec
	Bus type	PCI
	Multiple block instance support	No
	Multiple board support	Yes

# Contec PIO-32/32T(PCI) Digital Input

---

**Purpose** PIO-32/32T(PCI) Digital Input block

**Library** xPC Target Library for Contec

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

## **Block Parameters**

### **I/O format**

Select `serial` or `parallel`. If you select `serial`, the block is configured to accept up to 32 one-bit input channels. If you select `parallel`, the block is configured to accept a single 32-bit input channel and the channel vector parameter is unavailable.

### **Channel vector**

If you selected `serial` I/O format, enter a vector of numbers to specify the input channels for serial I/O format. For example, to use the first and third digital input channels, enter

[1, 3]

The channel numbers can occur in any order, but the numbers must be in the range 1 to 32.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## Contec PIO-32/32T(PCI) Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The device ID of this board is 8152.

# Contec PIO-32/32T(PCI) Digital Output

---

**Purpose** PIO-32/32T(PCI) Digital Output block

**Library** xPC Target Library for Contec

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 TTL high

## **Block Parameters**

### **I/O format**

Select serial or parallel. If you select serial, the block is configured to accept up to 32 one-bit input channels for output. If you select parallel, the block is configured to accept a single 32-bit channel and the channel vector parameter is unavailable.

### **Channel vector**

For serial I/O format, enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1, 3]

The channel numbers can occur in any order, but the numbers must be in the range 1 to 32.

### **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 (0 or 1) for the output channels. Enter a scalar or a vector that is the

# Contec PIO-32/32T(PCI) Digital Output

---

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 selected parallel I/O format, the values can be in the form [hex2dec('ffffff')].

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

The device ID of this board is 8152.

# Contec PO-64L(PCI)H

---

**Board** Contec PO-64L(PCI)H

**General Description** The Contec PO-64L(PCI)H is a digital output board with 64 digital output channels. The supporting block can control up to 32 bits that you can configure as either a single 32-bit port (Single 32-bit Port mode) or up to 32 1-bit channels (32 1-bit Channels mode).

xPC Target supports this board with this driver block:

- Contec PO-64L(PCI)H Digital Output

<b>Board Characteristics</b>	Board name	PO-64L(PCI)H
	Manufacturer	Contec
	Bus type	PCI
	Multiple block instance support	Yes — for 32 1-bit Channels mode
	Multiple board support	Yes



# Contec PO-64L(PCI)H Digital Output

**Purpose** PO-64L(PCI)H Digital Output block

**Library** xPC Target Library for Contec

## Scaling Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double (**Format**:32 1-bit Channels)

uint32 (**Format**:Single 32-bit Port)

### Scaling

Double:  
< 0.5 = TTL low

>0.5 = TTL high

uint32:  
Bit clear = TTL low

Bit set = TTL high

## Block Parameters

### Format

From the list, select either

- **32 1-bit Channels** — Indicates that the channel vector specifies the configuration of the block. Each output is a double with value of 0 or 1. The value is 0 when the hardware input voltage is low.

With this mode, the block reads, modifies, then writes the data. This enables you to use multiple instances of this block, with each block addressing a different bit. However, for efficiency, use a single block for each 32-bit group. Use multiple block instances only if your application requires multiple accesses.

- **Single 32-bit Port** — Indicates that the output from the block is a 32-bit integer where all 32 bits on the board feed into the single output. The least significant bit is the lowest numbered bit in the Contec user manual.

# Contec PO-64L(PCI)H Digital Output

---

If you choose this mode, use the bit packing and unpacking blocks (Digital I/O Bit-Packing and Digital I/O Bit-Unpacking) to construct appropriate data frames.

## Group

From the list, select either

- A (Bits 0-31) — Selects the 32-bit word from Group A to address.
- B (Bits 32-63) — Selects the 32-bit word from Group B to address.

## Channel vector

(32 1-bit Channels mode only) Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1, 3]

This vector indexes into group A or B channels. For group A, channel vector [1, 3] specifies channels 0 and 2. For group B, channel vector [1, 3] specifies channels 32 and 34.

The channel numbers can occur in any order, but the numbers must be in the range 1 to 32.

## 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.

## Final action vector

The final action 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 -1, the block sets the channel to the value specified in the **Final value vector** value for that channel. If you specify a value of 0, the channel remains at the last value attained while the model was running.

### **Final value vector**

The final value vector contains the final value for each output channel. 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 the **Final action vector** is -1, the block sets the channel to this value on model termination.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# Data Translation

---

This chapter describes Data Translation I/O boards supported by xPC Target (<http://www.datx.com>).

## Data Translation DT2821

I/O board with 16 single-ended or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

## Data Translation DT2821-F-8DI

I/O board with 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

## Data Translation DT2821-G-8DI

I/O board with 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

## Data Translation DT2821-F-16SE

I/O board with 16 single-ended analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

Data Translation DT2821-G-8DI	I/O board with 16 single-ended analog input (A/D) channels, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.
Data Translation DT2823	I/O board 4 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.
Data Translation DT2824-PGH	I/O board with 16 single-ended or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.
Data Translation DT2824-PGL	I/O board with 16 single-ended or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.
Data Translation DT2825	I/O board with 16 single-ended or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.
Data Translation DT2827	I/O board with 4 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

---

Data Translation DT2828

I/O board with 4 single-ended analog input (A/D) channels, 2 analog output (D/A) channels, and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

“Boards and Blocks — Alphabetical List” on page 19-4

Description of block parameters for Data Translation driver blocks.

## **Boards and Blocks – Alphabetical List**



**Board** Data Translation DT2821

## General Description

The DT2821 is an I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 50 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target support supports this board with these driver blocks:

- Data Translation DT2821 Analog Input (A/D)
- Data Translation DT2821 Analog Output (D/A)
- Data Translation DT2821 Digital Input
- Data Translation DT2821 Digital Output

## Board Characteristics

Board Name	DT2821
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2821 Analog Input (A/D)

---

**Purpose** DT2821 Analog Input block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

If you choose Single-ended (16 channels) from the **Input coupling** list, enter numbers between 1 and 16. If you choose Differential (8 channels) from the **Input coupling** list, enter numbers between 1 and 8. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### **Gain vector**

Enter 1, 2, 4, or 8 for each of the channels in the **Channel vector** to specify the gain for that channel. The gain vector must be the same length as the **Channel vector**. (If you enter a scalar, it is automatically expanded to channel vector). This driver allows the gain of each channel to be different. The gain is applied prior to sampling the voltage.

For example, if you have an input signal from -1 to +1 volts, and a gain of 8, the signal is amplified -8 to +8 volts. Select a range equal or larger than the amplified voltage. For example, select a range of +- 10V. After the signal voltage is sampled, this block divides by the gain to output the original signal value.

# Data Translation DT2821 Analog Input (A/D)

---

## **Range**

From the list, choose either +- 10V (-10 volts to +10 volts), or 0 - 10V (0 volts to +10 volts). This specifies the effective range which is the same for all channels and must correspond with the input range setting on the board.

## **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

This choice must correspond to the input mode setting on the board.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2821 Analog Output (D/A)

---

**Purpose** DT2821 Analog Output block

**Library** xPC Target Library for Data Translation

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the **Channel vector**. The range vector must be the same length as the **Channel vector**. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5

# Data Translation DT2821 Analog Output (D/A)

---

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

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the Output Range Selection settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2821 Digital Input

---

**Purpose** DT2821 Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2821 series boards have two I/O ports, each containing 8 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.

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

## **Block Parameters**

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.

# Data Translation DT2821 Digital Input

---

## **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

# Data Translation DT2821 Digital Output

---

**Purpose** DT2821 Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2821 series boards have two I/O ports, each containing 8 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, that port is configured for output.

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## **Block Parameters**

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.



# Data Translation DT2821 Digital Output

---

## **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

# Data Translation DT2821-F-8DI

---

## Board

Data Translation DT2821-F-8DI

## General Description

The DT2821-F-8DI is an I/O board with 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 150 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2821-F-8DI Analog Input (A/D)
- Data Translation DT2821-F-8DI Analog Output (D/A)
- Data Translation DT2821-F-8DI Digital Input
- Data Translation DT2821-F-8DI Digital Output

## Board Characteristics

Board Name	DT2821-F-8DI
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2821-F-8DI Analog Input (A/D)

**Purpose** DT2821-F-8DI Analog Input block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### **Gain vector**

Enter 1, 2, 4, or 8 for each of the channels in the **Channel vector** to specify the gain for that channel. The gain vector must be the same length as the **Channel vector**. (If you enter a scalar, it is automatically expanded to channel vector). This driver allows the gain of each channel to be different. The gain is applied prior to sampling the voltage.

For example, if you have an input signal from -1 to +1 volts, and a gain of 8, the signal is amplified -8 to +8 volts. Select a range equal or larger than the amplified voltage. For example, select a range of +- 10V. After the signal voltage is sampled, this block divides by the gain to output the original signal value.

### **Range**

From the list, choose either +- 10V (-10 volts to +10 volts), or 0-10V (0 volts to +10 volts). This specifies the effective range which is the same for all channels and must correspond with the input range setting on the board.

# Data Translation DT2821-F-8DI Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2821-F-8DI Analog Output (D/A)

**Purpose** DT2821-F-8DI Analog Output block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the **Channel vector**. The range vector must be the same length as the **Channel vector**. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5

# Data Translation DT2821-F-8DI Analog Output (D/A)

---

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

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the Output Range Selection settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2821-F-8DI Digital Input

**Purpose** DT2821-F-8DI Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2821-F-8DI series boards have two I/O ports, each containing 8 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.

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

## **Block Parameters**

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.

# Data Translation DT2821-F-8DI Digital Input

---

## **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



# Data Translation DT2821-F-8DI Digital Output

**Purpose** DT2821-F-8DI Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2821-F-8DI series boards have two I/O ports, each containing 8 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, that port is configured for output.

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## **Block Parameters**

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2821-F-8DI Digital Output

---

## **Base address**

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

0x300

## Board

Data Translation DT2821-G-8DI

## General Description

The DT2821-G-8DI is an I/O board with 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 250 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2821-G-8DI Analog Input (A/D)
- Data Translation DT2821-G-8DI Analog Output (D/A)
- Data Translation DT2821-F-8DI Digital Input
- Data Translation DT2821-F-8DI Digital Output

## Board Characteristics

Board Name	DT2821-G-8DI
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2821-G-8DI Analog Input (A/D)

---

**Purpose** DT2821-G-8DI Analog Input block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### **Gain vector**

Enter 1, 2, 4, or 8 for each of the channels in the **Channel vector** to specify the gain for that channel. The gain vector must be the same length as the **Channel vector**. (If you enter a scalar, it is automatically expanded to channel vector). This driver allows the gain of each channel to be different. The gain is applied prior to sampling the voltage.

For example, if you have an input signal from -1 to +1 volts, and a gain of 8, the signal is amplified -8 to +8 volts. Select a range equal or larger than the amplified voltage. For example, select a range of +-10V. After the signal voltage is sampled, this block divides by the gain to output the original signal value.

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V (-5 volts to +5 volts), or 0-10V (0 volts to +10 volts). This specifies the effective range which is the same for all channels and must correspond with the input range setting on the board.

# Data Translation DT2821-G-8DI Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2821-G-8DI Analog Output (D/A)

**Purpose** DT2821-G-8DI Analog Output block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 2. 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**

Enter a range code for each of the channels in the **Channel vector**. The range vector must be the same length as the **Channel vector**. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +5	5
0 to +10	10

# Data Translation DT2821-G-8DI Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the Output Range Selection settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2821-G-8DI Digital Input

---

**Purpose** DT2821-G-8DI Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2821-G-8DI series boards have two I/O ports, each containing 8 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.

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

**Block Parameters**

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.



# Data Translation DT2821-G-8DI Digital Input

---

## **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

# Data Translation DT2821-G-8DI Digital Output

---

**Purpose** DT2821-G-8DI Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2821-G-8DI series boards have two I/O ports, each containing 8 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, that port is configured for output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2821-G-8DI Digital Output

---

## **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

# Data Translation DT2821-F-16SE

---

**Board** Data Translation DT2821-F-16SE

**General Description**

The DT2821-F-16SE is an I/O board with 16 single-ended analog input (A/D) channels (12-bit) with a maximum sample rate of 150 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2821-F-16SE Analog Input (A/D)
- Data Translation DT2821-F-16SE Analog Output (D/A)
- Data Translation DT2821-F-16SE Digital Input
- Data Translation DT2821-F-16SE Digital Output

**Board Characteristics**

Board Name	DT2821-F-16SE
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2821-F-16SE Analog Input (A/D)

**Purpose** DT2821-F-16SE Analog Input block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### **Gain vector**

Enter 1, 2, 4, or 8 for each of the channels in the **Channel vector** to specify the gain for that channel. The gain vector must be the same length as the **Channel vector**. (If you enter a scalar, it is automatically expanded to channel vector). This driver allows the gain of each channel to be different. The gain is applied prior to sampling the voltage.

For example, if you have an input signal from -1 to +1 volts, and a gain of 8, the signal is amplified -8 to +8 volts. Select a range equal or larger than the amplified voltage. For example, select a range of +- 10V. After the signal voltage is sampled, this block divides by the gain to output the original signal value.

### **Range**

From the list, choose either +- 10V (-10 volts to +10 volts), or 0-10V (0 volts to +10 volts). This specifies the effective range which is the same for all channels and must correspond with the input range setting on the board.

# Data Translation DT2821-F-16SE Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2821-F-16SE Analog Output (D/A)

**Purpose** DT2821-F-16SE Analog Output block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the **Channel vector**. The range vector must be the same length as the **Channel vector**. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +5	5
0 to +10	10

# Data Translation DT2821-F-16SE Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the Output Range Selection settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



# Data Translation DT2821-F-16SE Digital Input

**Purpose** DT2821-F-16SE Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2821-F-16SE series boards have two I/O ports, each containing 8 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 of Input to Output

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

## Block Parameters

**Port**  
From the list, choose 1 or 2.

### Channel vector

Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.

# Data Translation DT2821-F-16SE Digital Input

---

## **Base address**

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

0x300

# Data Translation DT2821-F-16SE Digital Output

**Purpose** DT2821-F-16SE Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2821-F-16SE series boards have two I/O ports, each containing 8 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, that port is configured for output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

**Port**  
From the list, choose 1 or 2.

### Channel vector

Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2821-F-16SE Digital Output

---

## **Base address**

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

0x300

## Board

Data Translation DT2821-G-16SE

## General Description

The DT2821-G-16SE is an I/O board with 16 single-ended analog input (A/D) channels (12-bit) with a maximum sample rate of 250 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2821-G-16SE Analog Input (A/D)
- Data Translation DT2821-G-16SE Analog Output (D/A)
- Data Translation DT2821-G-16SE Digital Input
- Data Translation DT2821-G-16SE Digital Output

## Board Characteristics

Board Name	DT2821-G-16SE
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2821-G-16SE Analog Input (A/D)

---

**Purpose** DT2821-G-16SE Analog Input block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### **Gain vector**

Enter 1, 2, 4, or 8 for each of the channels in the **Channel vector** to specify the gain for that channel. The gain vector must be the same length as the **Channel vector**. (If you enter a scalar, it is automatically expanded to channel vector). This driver allows the gain of each channel to be different. The gain is applied prior to sampling the voltage.

For example, if you have an input signal from -1 to +1 volts, and a gain of 8, the signal is amplified -8 to +8 volts. Select a range equal or larger than the amplified voltage. For example, select a range of **+ -10V**. After the signal voltage is sampled, this block divides by the gain to output the original signal value.

### **Range**

From the list, choose either + -10V (-10 volts to +10 volts), + -5V (-5 volts to +5 volts), or 0 -10V (0 volts to +10 volts). This specifies the effective range which is the same for all channels and must correspond with the input range setting on the board.

# Data Translation DT2821-G-16SE Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2821-G-16SE Analog Output (D/A)

**Purpose** DT2821-G-16SE Analog Output block

**Library** xPC Target Library for Data Translation

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the **Channel vector**. The range vector must be the same length as the **Channel vector**. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5



# Data Translation DT2821-G-16SE Analog Output (D/A)

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

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the Output Range Selection settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2821-G-16SE Digital Input

---

**Purpose** DT2821-G-16SE Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2821-G-16SE series boards have two I/O ports, each containing 8 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.

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

**Block Parameters**

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.

# Data Translation DT2821-G-16SE Digital Input

---

## **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

# Data Translation DT2821-G-16SE Digital Output

---

**Purpose** DT2821-G-16SE Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2821-G-16SE series boards have two I/O ports, each containing 8 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, that port is configured for output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2821-G-16SE Digital Output

---

## **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

# Data Translation DT2823

---

## Board

Data Translation DT2823

## General Description

The DT2823 is an I/O board 4 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 100 kHz, 2 analog output (D/A) channels (16-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2823 Analog Input (A/D)
- Data Translation DT2823 Analog Output (D/A)
- Data Translation DT2823 Digital Input
- Data Translation DT2823 Digital Output

## Board Characteristics

Board Name	DT2823
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2823 Analog Input (A/D)

---

**Purpose** DT2823 Analog Input block

**Library** xPC Target Library for Data Translation

**Note** The range for the DT2823 is -10 to + 10 volts.

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 4. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2823 Analog Output (D/A)

**Purpose** DT2823 Analog Output block

**Library** xPC Target Library for Data Translation

**Note** The range of the DT2823 is -10 to +10 volts.

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the **Channel vector**. The range vector must be the same length as the **Channel vector**. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5



# Data Translation DT2823 Analog Output (D/A)

---

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

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the Output Range Selection settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

### **Sample time**

Base sample time of a multiple of the base sample time.

### **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2823 Digital Input

---

**Purpose** DT2823 Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2823 series boards have two I/O ports, each containing 8 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.

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

**Block Parameters** **Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.

# Data Translation DT2823 Digital Input

---

## **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

# Data Translation DT2823 Digital Output

---

**Purpose** DT2823 Digital Output block

**Library** xPC Target for Data Translation

**Note** DT2823 series boards have two I/O ports, each containing 8 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, that port is configured for output.

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

**Block Parameters**

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2823 Digital Output

---

## **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

# Data Translation DT2824-PGH

---

**Board** Data Translation DT2824-PGH

**General Description** The DT2824-PGH is an I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 50 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2824-PGH Analog Input (A/D)
- Data Translation DT2824-PGH Digital Input
- Data Translation DT2824-PGH Digital Output

**Board Characteristics**

Board Name	DT2824-PGH
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2824-PGH Analog Input (A/D)

**Purpose** DT2824-PGH Analog Input block

**Library** xPC Target Library for Data Translation

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

If you choose Single-ended (16 channels) from the **Input coupling** list, enter numbers between 1 and 16. If you choose Differential (8 channels) from the **Input coupling** list, then enter numbers between 1 and 8. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### Gain vector

Enter 1, 2, 4, or 8 for each of the channels in the **Channel vector** to specify the gain for that channel. The gain vector must be the same length as the **Channel vector**. (If you enter a scalar, it is automatically expanded to channel vector). This driver allows the gain of each channel to be different. The gain is applied prior to sampling the voltage.

For example, if you have an input signal from -1 to +1 volts, and a gain of 8, the signal is amplified -8 to +8 volts. Select a range equal or larger than the amplified voltage. For example, select a range of +- 10V. After the signal voltage is sampled, this block divides by the gain to output the original signal value.

# Data Translation DT2824-PGH Analog Input (A/D)

---

## **Range**

From the list, choose either +- 10V (-10 volts to +10 volts), or 0-10V (0 volts to +10 volts). This specifies the effective range which is the same for all channels and must correspond with the input range setting on the board.

## **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- differential channels (8 channels)

This choice must correspond to the input mode setting on the board.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

For example, if the first channel is -10 to +10 volts, and the second channel is 0 to 5 volts, enter

[-10,5]



# Data Translation DT2824-PGH Digital Input

**Purpose** DT2824-PGH Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2824-PGH series boards have two I/O ports, each containing 8 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.

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

**Block Parameters**

**Port** From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.

# Data Translation DT2824-PGH Digital Input

---

## **Base address**

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

0x300

# Data Translation DT2824-PGH Digital Output

**Purpose** DT2824-PGH Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2824-PGH series boards have two I/O ports, each containing 8 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, that port is configured for output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

**Port**  
From the list, choose 1 or 2.

### Channel vector

Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2824-PGH Digital Output

---

## **Base address**

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

0x300

**Board** Data Translation DT2824-PGL

**General Description**

The DT2824-PGL is an I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 50 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2824-PGL Analog Input (A/D)
- Data Translation DT2824-PGL Digital Input
- Data Translation DT2824-PGL Digital Output

**Board Characteristics**

Board Name	DT2824-PGL
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2824-PGL Analog Input (A/D)

---

**Purpose** DT2824-PGL Analog Input block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

If you choose Single-ended (16 channels) from the **Input coupling** list, enter numbers between 1 and 16. If you choose Differential (8 channels) from the **Input coupling** list, then enter numbers between 1 and 8. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### **Gain vector**

Enter 1, 10, 100, or 500 for each of the channels in the **Channel vector** to specify the gain for that channel. The gain vector must be the same length as the **Channel vector**. (If you enter a scalar, it is automatically expanded to channel vector). This driver allows the gain of each channel to be different. The gain is applied prior to sampling the voltage.

For example, if you have an input signal from -1 to +1 volts, and a gain of 8, the signal is amplified -8 to +8 volts. Select a range equal or larger than the amplified voltage. For example, select a range of +- 10V. After the signal voltage is sampled, this block divides by the gain to output the original signal value.

### **Range**

From the list, choose either +- 10V (-10 volts to +10 volts), or 0-10V (0 volts to +10 volts). This specifies the effective range which is

# Data Translation DT2824-PGL Analog Input (A/D)

---

the same for all channels and must correspond with the input range setting on the board.

## **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- differential channels (8 channels)

This choice must correspond to the input mode setting on the board.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

For example, if the first channel is -10 to +10 volts, and the second channel is 0 to 5 volts, enter

[-10,5]

# Data Translation DT2824-PGL Digital Input

---

**Purpose** DT2824-PGL Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2824-PGL series boards have two I/O ports, each containing 8 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 of Input to Output

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

## Block Parameters

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.



# Data Translation DT2824-PGL Digital Input

---

## **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

# Data Translation DT2824-PGL Digital Output

---

**Purpose** DT2824-PGL Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2824-PGL series boards have two I/O ports, each containing 8 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, that port is configured for output.

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## **Block Parameters**

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2824-PGL Digital Output

---

## **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

# Data Translation DT2825

---

## Board

Data Translation DT2825

## General Description

The DT2825 is an I/O board with 16 single-ended or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 45 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2825 Analog Input (A/D)
- Data Translation DT2825 Analog Output (D/A)
- Data Translation DT2825 Digital Input
- Data Translation DT2825 Digital Output

## Board Characteristics

Board Name	DT2825
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2825 Analog Input (A/D)

**Purpose** DT2825 Analog Input block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

If you choose Single-ended (16 channels) from the **Input coupling** list, enter numbers between 1 and 16. If you choose Differential (8 channels) from the **Input coupling** list, then enter numbers between 1 and 8. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### **Gain vector**

Enter 1, 10, 100, or 500 for each of the channels in the **Channel vector** to specify the gain for that channel. The gain vector must be the same length as the **Channel vector**. (If you enter a scalar, it is automatically expanded to channel vector). This driver allows the gain of each channel to be different. The gain is applied prior to sampling the voltage.

For example, if you have an input signal from -1 to +1 volts, and a gain of 8, the signal is amplified -8 to +8 volts. Select a range equal or larger than the amplified voltage. For example, select a range of +-10V. After the signal voltage is sampled, this block divides by the gain to output the original signal value.

# Data Translation DT2825 Analog Input (A/D)

---

## **Range**

From the list, choose either  $\pm 10V$  (-10 volts to +10 volts), or 0-10V (0 volts to +10 volts). This specifies the effective range which is the same for all channels and must correspond with the input range setting on the board.

## **Input coupling**

From the list, choose one of the input modes:

- Single-ended channels (16 channels)
- differential channels (8 channels)

This choice must correspond to the input mode setting on the board.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2825 Analog Output (D/A)

**Purpose** DT2825 Analog Output block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the **Channel vector**. The range vector must be the same length as the **Channel vector**. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +5	5
0 to +10	10

# Data Translation DT2825 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings have to correspond to the Output Range Selection settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



# Data Translation DT2825 Digital Input

**Purpose** DT2825 Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2825 series boards have two I/O ports, each containing 8 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 of Input to Output

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

## Block Parameters

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.

# Data Translation DT2825 Digital Input

---

## **Base address**

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

0x300

# Data Translation DT2825 Digital Output

**Purpose** DT2825 Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2825 series boards have two I/O ports, each containing 8 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, that port is configured for output.

## Scaling of Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

$<0.5 = \text{TTL low}$   $\geq 0.5 = \text{TTL high}$

## Block Parameters

### Port

From the list, choose 1 or 2.

### Channel vector

Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2825 Digital Output

---

## **Base address**

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

0x300

## Board

Data Translation DT2827

## General Description

The DT2827 is an I/O board with 4 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 100 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2827 Analog Input (A/D)
- Data Translation DT2827 Analog Output (D/A)
- Data Translation DT2827 Digital Input
- Data Translation DT2827 Digital Output

## Board Characteristics

Board Name	DT2827
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2827 Analog Input (A/D)

---

**Purpose** DT2827 Analog Input block

**Library** xPC Target Library for Data Translation

**Note** The range for this board is -10 to +10 volts.

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

**Block Parameters**

**Channel vector**  
Enter numbers between 1 and 4. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

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

**Base address**  
Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2827 Analog Output (D/A)

- Purpose** DT2827 Analog Output block
- Library** xPC Target Library for Data Translation
- Note** The range for this board is -10 to + 10 volts.

Scaling of Input to Output	Hardware Output	Block Input Data Type	Scaling
	Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 2. 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 vector

Enter a range code for each of the channels in the **Channel vector**. The range vector must be the same length as the **Channel vector**. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5

# Data Translation DT2827 Analog Output (D/A)

---

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

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the Output Range Selection settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



# Data Translation DT2827 Digital Input

**Purpose** DT2827 Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2827 series boards have two I/O ports, each containing 8 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 of Input to Output

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

## Block Parameters

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.

# Data Translation DT2827 Digital Input

---

## **Base address**

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

0x300

# Data Translation DT2827 Digital Output

**Purpose** DT2827 Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2827 series boards have two I/O ports, each containing 8 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, that port is configured for output.

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## **Block Parameters**

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2827 Digital Output

---

## **Base address**

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

0x300

## Board

Data Translation DT2828

## General Description

The DT2828 is an I/O board with 4 single-ended analog input (A/D) channels (12-bit) with a maximum sample rate of 100 kHz, 2 analog output (D/A) channels (12-bit), and 16 digital I/O lines that can be configured in groups of 8 for either input or output.

xPC Target supports this board with these driver blocks:

- Data Translation DT2828 Analog Input (A/D)
- Data Translation DT2828 Analog Output (D/A)
- Data Translation DT2828 Digital Input
- Data Translation DT2828 Digital Output

## Board Characteristics

Board Name	DT2828
Manufacturer	Data Translation
Bus Type	ISA
Access Method	I/O mapped
Multiple block instance support	A/D:No, D/A:No, Digital I/O:Yes
Multiple board support	Yes

# Data Translation DT2828 Analog Input (A/D)

---

**Purpose** DT2828 Analog Input block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 4. For example, to use the first and third analog output (A/D) channels, enter

[1,3]

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

### **Gain vector**

Enter 1, 2, 4, or 8 for each of the channels in the **Channel vector** to specify the gain for that channel. The gain vector must be the same length as the **Channel vector**. (If you enter a scalar, it is automatically expanded to channel vector). This driver allows the gain of each channel to be different. The gain is applied prior to sampling the voltage.

For example, if you have an input signal from -1 to +1 volts, and a gain of 8, the signal is amplified -8 to +8 volts. Select a range equal or larger than the amplified voltage. For example, select a range of **+10V**. After the signal voltage is sampled, this block divides by the gain to output the original signal value.

### **Range**

From the list, choose either **+10V** (-10 volts to +10 volts), or **0-10V** (0 volts to +10 volts). This specifies the effective range which is the same for all channels and must correspond with the input range setting on the board.

# Data Translation DT2828 Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2828 Analog Output (D/A)

**Purpose** DT2828 Analog Output block

**Library** xPC Target Library for Data Translation

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 2. 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 vector**

Enter a range code for each of the channels in the **Channel vector**. The range vector must be the same length as the **Channel vector**. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +5	5
0 to +10	10



# Data Translation DT2828 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the Output Range Selection settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Data Translation DT2828 Digital Input

---

**Purpose** DT2828 Digital Input block

**Library** xPC Target Library for Data Translation

**Note** DT2828 series boards have two I/O ports, each containing 8 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 of Input to Output

### Hardware Input

TTL

### Block Output Data Type

Double

### Scaling

TTL low = 0.0  
TTL high = 1.0

## Block Parameters

### Port

From the list, choose 1 or 2.

### Channel vector

Enter numbers between 1 and 8 to select the digital input lines you use 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 you use.

For example, to use all of the digital inputs for this 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.

# Data Translation DT2828 Digital Input

---

## **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

# Data Translation DT2828 Digital Output

---

**Purpose** DT2828 Digital Output block

**Library** xPC Target Library for Data Translation

**Note** DT2828 series boards have two I/O ports, each containing 8 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, that port is configured for output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## Block Parameters

**Port**  
From the list, choose 1 or 2.

**Channel vector**  
Enter numbers between 1 and 8. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of digital lines you use.

For example, to use all of the digital outputs for this 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.

# Data Translation DT2828 Digital Output

---

## **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

---

This chapter describes Diamond Systems I/O boards supported by xPC Target (<http://www.diamondsystems.com>). To read about the Diamond Systems serial communications boards supported by xPC target, see Chapter 2, “Serial Communications Support”.

Diamond Diamond-MM	DAS16-compatible I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, 8 digital input lines, and 8 digital output lines.
Diamond Diamond-MM-16-AT	PC/104 I/O board with 16 single-ended or 8 differential analog input (A/D) channels (16-bit), 4 optional analog output (D/A) channels (12-bit), and 8 digital input and output lines.
Diamond Diamond-MM-32-AT	PC/104 I/O board with 32 single or 16 differential analog input (A/D) channels, 4 analog output (D/A) channels, and 24 digital input and output lines.
Diamond Garnet-MM	I/O board with 24 or 48 high current digital I/O lines that can be configured in groups of 8 for either digital input or digital output.

Diamond Onyx-MM	I/O board with 48 digital I/O lines that can be configured in groups of 8 for either digital input or digital output, counters, and timers.
Diamond Onyx-MM-DIO	I/O board with 48 digital I/O lines that can be configured in groups of 8 for either digital input or digital output.
Diamond Prometheus and Athena	Intel 486-based embedded PC/104 CPU board with 4 serial ports, 2 USB ports, 1 parallel port, keyboard and mouse ports, floppy and IDE driver connectors, a 100BaseT Ethernet connector, and provision for solid state flashdisk modules.
Diamond Quartz-MM-5	8 digital input lines, 8 digital output lines, and 10 counter/timers.
Diamond Quartz-MM-10	8 digital input line, 8 digital output lines, and 10 counter/timers.
Diamond Ruby-MM	PC/104 I/O board with 4 or 8 single analog output (D/A) channels, unipolar and bipolar operation, +/- 10V, +/- 5V, 0-10V, 0-5V fixed ranges, +/- 2.5V, 0-2.5V user-adjustable ranges, and 24 digital input and output lines.
Diamond Ruby-MM-416	Four 16-bit analog output (D/A) channels, and 24 digital I/O lines that can be configured in groups of 8 for either input or output.
Diamond Ruby-MM-1612	Sixteen 12-bit analog output (D/A) channels, and 24 digital I/O lines which can be configured in groups of 8 for either input or output.
“Boards and Blocks — Alphabetical List” on page 20-3	Description of block parameters for Diamond Systems driver blocks.



## **Boards and Blocks – Alphabetical List**

# Diamond Diamond-MM

---

## Board

Diamond Diamond-MM

## General Description

The Diamond-MM is a DAS16 compatible I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 100 kHz, 2 analog output (D/A) channels (12-bit), 8 digital input lines, and 8 digital output lines.

xPC Target supports this board with these driver blocks:

- Diamond MM Analog Input (A/D)
- Diamond MM Analog Output (D/A)
- Diamond MM Digital Input
- Diamond MM Digital Output

## Board Characteristics

Board name	Diamond-MM
Manufacturer	Diamond Systems Corporation
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Diamond MM Analog Input (A/D)

**Purpose** MM Analog Input block

**Library** xPC Target Library for Diamond

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

## Block Parameters

### Number of channels

If you select 16 channels (**Coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of single A/D channels used. If you select eight channels (**Coupling** parameter set to 8 differential channels) enter a number between 1 and 8.

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

Note, you cannot select the starting channel for this block, you can only select the number of channels. This block works differently from the Diamond-MM-32-AT A/D block, where you can specify the starting channel.

### Range

Enter an input range code for all A/D channels. This driver does not allow the selection of a different range for each channel. The input range is the same for all A/D channels.

The following table is a list of the ranges for this driver and the corresponding range codes.

# Diamond MM Analog Input (A/D)

Input Range (V)	Range Code	Input Range (V)	Range Code
-10 to +10	-10	0 to +10	10
-5 to +5	-5	0 to +5	5
-2.5 to + 2.5	-2.5	0 to +2	2
-1 to +1	-1	0 to +1	1
-0.5 to +0.5	-5		

The gain jumpers on the board have to be in the correct positions for the chosen range. The bipolar jumper on the board has to be in the bipolar position, if a bipolar range is used or in the unipolar position, when a unipolar range is used.

## Coupling

From the list, select one from the following list of input modes:

- 16 single-ended channels
- 8 differential channels

This choice must correspond to the jumper setting in block J6 on the board.

## Show error status output (E)

Select this check box to add a port labeled **E** to the block and to display (the contents of) that special port. This output will always have a value of 0 unless a problem is detected while attempting an A/D conversion. In the unlikely event that an error occurs, the port has a nonzero value. This nonzero value takes the form of a real number whose binary representation of 1's and 0's (true and false) indicates which channels have errors.

## Sample time

Enter the base sample time or a multiple of the base sample time.

# Diamond MM Analog Input (A/D)

---

## **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 Analog Output (D/A)

---

**Purpose** MM Analog Output block

**Library** xPC Target Library for Diamond

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

## Block Parameters

### Channel vector

This parameter is a combined Channel vector and Range vector. The number of elements defines the number of D/A channels used.

### Range vector

Enter a range code for each of the D/A channels used. This driver allows a different range for each channel with a maximum of 2 channels.

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 have to be in the correct positions for the ranges entered.

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

# Diamond MM Analog Output (D/A)

---

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 input 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 an out-of-range value for a channel, that value is adjusted to be within the correct range, as defined in the parameter **Range**.

## **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 Digital Input

---

**Purpose** MM Digital Input block

**Library** xPC Target Library for Diamond

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8. This driver allows the selection of individual digital input channels in any order. The number of elements defines the number of digital input channels you use. For example, to use all the digital input channels, enter

[1:8]

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



**Purpose** MM Digital Output block

**Library** xPC Target Library for Diamond

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 8. This driver allows the selection of individual digital input channels in any order. The number of elements defines the number of digital input channels you use. For example, to use all the digital output channels, enter

[ 1:8 ]

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

### 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 input 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 an out-of-range value for a

# Diamond MM Digital Output

---

channel, that value is adjusted to be within the correct range, as defined in the parameter **Range**.

## **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

**Board** Diamond Diamond-MM-16-AT

**General Description**

The Diamond MM-AT is a PC104 I/O board with 16 single-ended or 8 differential analog input (A/D) channels (16-bit), 4 optional analog output (D/A) channels (12-bit), 8 digital input and output lines.

Diamond-MM-16-AT boards have an 8-bit digital input port and an 8-bit digital output port.

xPC Target supports this board with these driver blocks:

- Diamond MM-16-AT Analog Input (A/D)
- Diamond MM-16-AT Analog Output (D/A)
- Diamond MM-16-AT Digital Input
- Diamond MM-16-AT Digital Output

xPC Target does not support the counters/timers on this board.

**Board Characteristics**

Board name	Diamond-MM-16-AT
Manufacturer	Diamond Systems Corporation
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Diamond MM-16-AT Analog Input (A/D)

---

**Purpose** MM-16-AT Analog Input block

**Library** xPC Target Library for Diamond

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

## **Block Parameters**

### **First Channel**

Enter the number of the first channel in a set of contiguous analog input channels. Depending on the channel configuration selected, the first channel number must be within the range 1 through 8 (**Coupling** parameter set to 8 differential channels) or 1 through 16 (**Coupling** parameter set to 16 single-ended channels).

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

### **Number of Channels**

Enter the number of input channels you want to use. The maximum number of channels varies between 1 and 16 and depends on the values of **Coupling** and the **First channel number**.

### **Range**

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

### **Coupling**

From the list, select one from the following list of input modes:

- Single-ended (16 channels)
- Differential (8 channels)

This choice must correspond to the jumper setting in block J4 on the board.

# Diamond MM-16-AT Analog Input (A/D)

---

## **Show error status output (E)**

Select this check box to add a port labeled **E** to the block and to display (the contents of) that special port. This output will always have a value of 0 unless a problem is detected while attempting an A/D conversion. In the unlikely event that an error occurs, the port has a nonzero value. This nonzero value takes the form of a real number whose binary representation of 1's and 0's (true and false) indicates which channels have errors.

## **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 base address DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Diamond MM-16-AT Analog Output (D/A)

---

**Purpose** MM-16-AT Analog Output block

**Library** xPC Target Library for Diamond

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

## **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, select 0 to 5V or -5V to 5V as the input voltage range of the board. The input range applies to all channels.

This choice must correspond to the jumper setting in block J5 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.

# Diamond MM-16-AT Analog Output (D/A)

---

## **Initial value vector**

The initial value vector contains the initial voltage values for the input 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 an out-of-range value for a channel, that value is adjusted to be within the correct range, as defined in the parameter **Range**.

## **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 base address DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Diamond MM-16-AT Digital Input

---

**Purpose** MM-16-AT Digital Input block

**Library** xPC Target Library for Diamond

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8. This driver allows the selection of individual digital input channels in any order. The number of elements defines the number of digital input channels you use. For example, to use all the digital input channels, enter

[1:8]

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

### **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 base address DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



# Diamond MM-16-AT Digital Output

**Purpose** MM-16-AT Digital Output block

**Library** xPC Target Library for Diamond

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8. This driver allows the selection of individual digital output channels in any order. The number of elements defines the number of digital output channels you use. For example, to use all of the digital output channels, enter

[ 1:8 ]

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

### **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 an out-of-range value for a

# Diamond MM-16-AT Digital Output

---

channel, that value is adjusted to be within the correct range, as defined in the parameter **Range**.

## **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 base address DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

**Board** Diamond Diamond-MM-32-AT

**General Description**

The Diamond-MM-32-AT is a PC104 I/O board with 32 single or 16 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 200 kHz, 4 analog output (D/A) channels (12-bit), 24 digital input and output lines.

xPC Target supports this board with these driver blocks:

- Diamond MM-32-AT Analog Input (A/D)
- Diamond MM-32-AT Frame Analog Input (A/D)
- Diamond MM-32-AT Analog Output (D/A)
- Diamond MM-32-AT Digital Input
- Diamond MM-32-AT Digital Output

**Board Characteristics**

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

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

---

**Purpose** MM-32-AT Analog Input block

**Library** xPC Target Library for Diamond

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

## **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 value of the **Channel Configuration** parameter, the first channel number must be within the range 1 through 32, 1 through 16, or 1 through 24.

### **Number of channels**

Enter the number of input channels you want to use. The maximum number of channels varies between 1 and 32 and

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

---

depends on **Channel configuration** and the **First channel number**.

## **Range**

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

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

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

0x300

# Diamond MM-32-AT Frame Analog Input (A/D)

---

**Purpose** MM-32-AT Frame Analog Input block

**Library** xPC Target Library for Diamond

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

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

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

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

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

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

Note, after you add this block to a model and are ready to configure the model, edit the xPC Target code generation options.

- 1** From the model, select **Simulation > Configuration Parameters**.
- 2** Select the **Real-Time Workshop** node.
- 3** In the **Target selection** section, from the **System target file** list, browse to and select `xpctarget.tlc`.
- 4** In the **xPC Target options** node, from the **I/O board generating the interrupt** list, select the value `Diamond-MM-32`. This specifies that the Diamond-MM-32-AT board generates the interrupt.

# Diamond MM-32-AT Frame Analog Input (A/D)

---

- 5 In the same node, from the **Real-time interrupt source** list, select the IRQ number you have jumpered on the board.
- 6 In the same node, for the **PCI slot/ISA base address** parameter, enter the same ISA address as for the Diamond-MM-32-AT Frame block **Base address** parameter.
- 7 Click **OK** and save the model.

---

**Note** To enable the interrupt for the frame-based block, the 16-bit bus jumper must be present in the board J7 jumper block. For further details, see the section labeled "16-Bit Bus" in the Diamond-MM-32-AT user's guide documentation (<http://www.diamondsystems.com/files/binaries/DMM32v2.64.pdf>).

---

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel configuration

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

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

# Diamond MM-32-AT Frame Analog Input (A/D)

---

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

## Output Signal Type

From the list, select either Vector or Frame:

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

## Range

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

## First channel number

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

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

## Number of channels

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



# Diamond MM-32-AT Frame Analog Input (A/D)

---

## **Number of scans per frame**

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

## **Interval between conversions within a scan**

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

## **Interval between scans**

Enter the interval, in seconds, between successive scans.

## **Base address**

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

0x300

# Diamond MM-32-AT Analog Output (D/A)

---

**Purpose** MM-32-AT Analog Output block

**Library** xPC Target Library for Diamond

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

## **Block Parameters**

### **Channel vector**

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

[1,2]

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

### **Range**

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

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

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

# Diamond MM-32-AT Analog Output (D/A)

---

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

[ 10, 10 ]

## **Reset vector**

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

## **Initial value vector**

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

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

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

0x300

# Diamond MM-32-AT Digital Input

---

**Purpose** MM-32-AT Digital Input block

**Library** xPC Target Library for Diamond

**Note** The Diamond-MM-32-AT has one 8255 chip with three ports (A,B,C). Each port has a maximum of 8 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 as input.

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

## **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.

# Diamond MM-32-AT Digital Input

---

## **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

---

**Purpose** MM-32-AT Digital Output block

**Library** xPC Target Library for Diamond

**Note** 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

## Block Parameters

### Channel vector

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

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

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

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

### Port

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

## **Reset vector**

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

## **Initial value vector**

The initial value vector contains the initial values (0 or 1) of the output channels. Enter a scalar or a vector that is the same length as the channel vector. If you enter a scalar, that value is used for all channels. The channels are set to these initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **Base address**

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

0x300

# Diamond Garnet-MM

---

## Board

Diamond Garnet-MM

## General Description

The Garnet-MM is an I/O board with 24 or 48 high current digital I/O lines that can be configured in groups of eight for either digital input or digital output. There are two versions of this board, 24 (GMM-24) or 48 (GMM-48) digital I/O lines. The 48 line version has two 82C55 chips. Each chip has three 8-bit I/O ports for a total of 48 lines. The 24 line version has one 82C55 chip with three 8-bit I/O ports for a total of 24 lines.

xPC Target supports this board with these driver blocks:

- Diamond Garnet-MM Digital Input
- Diamond Garnet-MM Digital Output

## Board Characteristics

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



# Diamond Garnet-MM Digital Input

**Purpose** Garnet-MM Digital Input block

**Library** xPC Target Library for Diamond

**Note** Each chip port of the Garnet-MM board 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 the port for output.

## Scaling Input to Output

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

## 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. Note, only select 1 for the 24 line version of the Garnet-MM board. Selecting 2 for the 24 line board has no effect. You can select either 1 or 2 for the 48 line version of the board.

# Diamond Garnet-MM Digital Input

---

**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

# Diamond Garnet-MM Digital Output

**Purpose** Garnet-MM Digital Output block

**Library** xPC Target Library for Diamond

Each chip port of the Garnet-MM board 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

## 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.

### 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.

# Diamond Garnet-MM Digital Output

---

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.

## **Chip**

From the list choose 1 or 2. Note, only select 1 for the 24 line version of the Garnet-MM board. Selecting 2 for the 24 line board has no effect. You can select either 1 or 2 for the 48 line version of the board.

## **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

**Board** Diamond Onyx-MM

**General Description**

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 Target does not support the Counter/Timer functionality of this board.

xPC Target supports this board with these driver blocks:

- Diamond Onyx-MM Digital Input
- Diamond Garnet-MM Digital Output

**Board Characteristics**

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

# Diamond Onyx-MM Digital Input

---

**Purpose** Onyx-MM Digital Input block

**Library** xPC Target Library for Diamond

**Note** 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.

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

## **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.

# Diamond Onyx-MM Digital Input

---

## **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

# Diamond Onyx-MM Digital Output

---

**Purpose** Onyx-MM Digital Output block

**Library** xPC Target Library for Diamond

**Note** 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.

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

## **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.

### **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.

## **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

# Diamond Onyx-MM-DIO

---

**Board** Diamond Onyx-MM-DIO

**General Description** The Onyx-MM is an I/O board with 48 digital I/O lines that can be configured in groups of eight for either digital input or digital output. xPC Target supports this board with these driver blocks:

- Diamond Onyx-MM Digital Input
- Diamond Onyx-MM Digital Output

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

# Diamond Onyx-MM-DIO Digital Input

**Purpose** Onyx-MM-DIO Digital Input block

**Library** xPC Target Library for Diamond

**Note** Onyx-MM-DIO 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.

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

## 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.

# Diamond Onyx-MM-DIO Digital Input

---

## **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

# Diamond Onyx-MM-DIO Digital Output

**Purpose** Onyx-MM-DIO Digital Output block

**Library** xPC Target Library for Diamond

**Note** Onyx-MM-DIO 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

## 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.

### 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

# Diamond Onyx-MM-DIO Digital Output

---

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.

## **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

# Diamond Prometheus and Athena

## Board

Diamond Prometheus and Athena

## General Description

xPC Target supports the Diamond Prometheus and Athena boards.

- **Diamond Prometheus** — This is an Intel 486-based embedded PC/104 CPU board with 4 serial ports, 2 USB ports, 1 parallel port, keyboard and mouse ports, floppy and IDE drive connectors, a 100BaseT Ethernet connector, and provision for solid-state flashdisk modules.

xPC Target supports Model PR-Z32-EA of Prometheus. In addition to the above functionality, Model PR-Z32-EA also contains a data acquisition subsystem. This subsystem supports 16 single-ended or 8 differential 16-bit A/D, 4 12-bit analog outputs, 24 programmable digital I/O channels, and a 16-bit counter/timer.

- **Diamond Athena** — This is a Pentium III class VIA Eden processor-based embedded PC/104 CPU with 4 serial ports, 4 USB ports, 1 parallel port, keyboard and mouse ports, floppy and IDE drive connectors, a 100BaseT Ethernet connector, and a 16-bit data acquisition circuit.

xPC Target supports the Prometheus Model PR-Z32-EA and Athena with four driver blocks:

- Diamond Prometheus/Athena Analog Input (A/D)
- Diamond Prometheus/Athena Analog Output (D/A)
- Diamond Prometheus/Athena Digital Input
- Diamond Prometheus/Athena Digital Output

xPC Target does not support the counter/timer on this board.

## Board Characteristics

Board names	Prometheus and Athena
Manufacturer	Diamond Systems
Bus type	PC/104

# Diamond Prometheus and Athena

---

Multiple block instance support      A/D: No, D/A: No, Digital I/O: Yes

Multiple board support                  No



# Diamond Prometheus/Athena Analog Input (A/D)

**Purpose** Prometheus/Athena Analog Input block

**Library** xPC Target Library for Diamond

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

## **Block Parameters**

### **First channel**

Enter the number of the first channel in a set of contiguous channels. Depending on the channel configuration selected, the first channel number must be within the range:

- 1 through 8, if the input coupling is differential
- 1 through 16, if the input coupling is single-ended

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

### **Number of channels**

Enter the number of input channels you want to use. The maximum number of channels varies between 1 and 16 and depends on the values of **First channel** and **Input coupling**. For example, if the value of **First channel** is 1 and **Input coupling** is Single-ended, the maximum value for **Number of channels** is 16.

### **Range (J13 setting)**

From the list, choose a voltage range. The input range applies to all channels. This range must agree with jumper settings in block J13 on the board.

### **Input coupling (J13 setting)**

From the list, select one from the following list of input modes:

- Single-ended (16 channels)
- Differential (8 channels)

# Diamond Prometheus/Athena Analog Input (A/D)

---

This choice must correspond to the jumper setting in block J13 on the board.

## **Show error status output port**

Select this check box to display real-time error information. This check box displays an output port labeled **E**. As long as no error is detected on any of the channels currently in use, this signal has a value of 0. Values of 1, 2, 4, 8, and so forth respectively indicate problems on channels 1, 2, 3, 4, and so forth. To use one signal to indicate errors on multiple channels, the driver combines these values. For example, the **E** value  $1 + 2 = 3$  encodes the concurrent errors on channels 1 and 2, the value  $2 + 4 = 6$  encodes the concurrent errors on channels 2 and 3, and so on.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the base address setting on the board. For example, if the base address is 280 (hexadecimal), enter

0x280

# Diamond Prometheus/Athena Analog Output (D/A)

**Purpose** Prometheus/Athena Analog Output block

**Library** xPC Target Library for Diamond

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

## **Block Parameters**

### **Channel Vector**

Enter a vector of 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**

From the list, choose a voltage range. The output range applies to all channels. This range must agree with jumper settings in block J13 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

# Diamond Prometheus/Athena Analog Output (D/A)

---

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 an out-of-range value for a channel, that value is adjusted to be within the correct range, as defined in the parameter **Range**.

## **Show error status output port**

Select this check box to display real-time error information. This check box displays an output port labeled **E**. As long as no error is detected on any of the channels currently in use, this signal has a value of 0. Values of 1, 2, 4, 8, and so forth respectively indicate problems on channels 1, 2, 3, 4, and so forth. To use one signal to indicate errors on multiple channels, the driver combines these values. For example, the **E** value  $1 + 2 = 3$  encodes the concurrent errors on channels 1 and 2, the value  $2 + 4 = 6$  encodes the concurrent errors on channels 2 and 3, and so on.

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry correspond to the DIP switch settings on the board. For example, if the base address is 280 (hexadecimal), enter

0x280

# Diamond Prometheus/Athena Digital Input

**Purpose** Prometheus/Athena Digital Input block

**Library** xPC Target Library for Diamond

**Note** Prometheus and Athena boards have three I/O ports, each containing eight digital I/O lines. You can independently configure these ports 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.

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

## Block Parameters

### Channel Vector

Enter a vector of numbers between 1 and 8 to select the digital input lines used from 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 of the digital inputs for the current port, enter

[1:8]

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

### Port

From the list, choose A, B, or C to choose one of the three I/O ports.

### Sample time

Enter a base sample time or a multiple of the base sample time

# Diamond Prometheus/Athena Digital Input

---

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 280 (hexadecimal), enter

0x280

# Diamond Prometheus/Athena Digital Output

**Purpose** Prometheus/Athena Digital Output block

**Library** xPC Target Library for Diamond

**Note** Prometheus and Athena boards have three I/O ports, each containing eight digital I/O lines. You can independently configure these ports for either input or output. Use a separate driver block for each port. By selecting the digital output driver block for a given port, that port is configured for output.

## Scaling Input to Output

### Hardware Input

TTL

### Block Output Data Type

Double

### Scaling

TTL low = 0.0

TTL high = 1.0

## Block Parameters

### Channel Vector

Enter a vector of numbers between 1 and 8 to select the digital output lines used from 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 of the digital outputs for the current port, enter

[1:8]

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

### Port

From the list, choose A, B, or C to choose one of the three I/O ports.

### 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

# Diamond Prometheus/Athena Digital Output

---

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 a base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 280 (hexadecimal), enter

0x280



## Board

Diamond Quartz-MM-10

## General Description

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

xPC Target supports this board with these driver blocks:

- Diamond Quartz-MM-10 Digital Input
- Diamond Quartz-MM-10 Digital Output
- Diamond Quartz-MM-10 Counter PWM
- Diamond Quartz-MM-10 Counter PWM & ARM
- Diamond Quartz-MM-10 Counter FM
- Diamond Quartz-MM-10 Counter FM & ARM
- Diamond Quartz-MM-10 PWM Capture
- Diamond Quartz-MM-10 FM Capture
- Diamond Quartz-MMxx

## Board Characteristics

Board name	Quartz-MM-10
Manufacturer	Diamond Systems Corporation
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	Yes — a channel can be accessed by at most one block
Multiple board support	Yes

# Diamond Quartz-MM-10 Digital Input

---

**Purpose** Quartz-MM-10 Digital Input block

**Library** xPC Target Library for Diamond

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

## **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

# Diamond Quartz-MM-10 Digital Output

**Purpose** Quartz-MM-10 Digital Output block

**Library** xPC Target Library for Diamond

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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

# Diamond Quartz-MM-10 Counter PWM

---

**Purpose** Quartz-MM-10 Counter PWM block

**Library** xPC Target Library for Diamond

**Note** The Quartz-MM10 has two AM9513A chips with five counters each. The Quartz-MM10 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.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, 4, or 5 to select which counter is used with this driver block. In each case, one block is needed for each counter.

### **Frequency base**

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### **Relative output frequency**

Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM signal.

For example, if the output frequency of a square wave must be 17.5 kHz, then choose F2=400kHz as the **Frequency base** and enter 0.04375 as the **Relative output frequency**.  $400 \text{ kHz} \times 0.04375 = 17.5 \text{ kHz}$

### **Level sequence of square wave**

From the list, choose either high-low or low-high:

# Diamond Quartz-MM-10 Counter PWM

---

- 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

# Diamond Quartz-MM-10 Counter PWM & ARM

---

**Purpose** Quartz-MM-10 Counter PWM & ARM block

**Library** xPC Target Library for Diamond

**Note** 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.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Duty cycle: Double Arm: Double	0 to 1 <0.5 disarmed ≥0.5 armed

**Block Parameters**

**Counter**  
From the list, choose 1, 2, 3, 4, or 5 to select which counter is used with this driver block. In each case, one block is needed for each counter.

**Frequency base**  
From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

**Relative output frequency**  
Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM signal.

For example, if the output frequency of a square wave must be 17.5 kHz, then choose F2=400kHz as the **Frequency base** and

# Diamond Quartz-MM-10 Counter PWM & ARM

---

enter 0.04375 as the **Relative output frequency**.  $400 \text{ kHz} \times 0.04375 = 17.5 \text{ kHz}$

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running. If the application is running, the second input port controls whether the counter is armed or disarmed. This parameter sets the TTL level when the counter is disarmed.

## **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle)

## **Base address**

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

0x300

# Diamond Quartz-MM-10 Counter FM

---

**Purpose** Quartz-MM-10 Counter FM block

**Library** xPC Target Library for Diamond

**Note** The Quartz-MM10 has two AM9513A chips with five counters.

The Quartz-MM10 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.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, 4, or 5 to select which counter is used with this driver block. In each case, one block is needed for each counter.

### **Frequency base**

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### **Output duty cycle**

Enter a value between 0 and 1 to set the duty cycle of the square wave. The duty cycle is held fixed during execution of the target application.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

### **Level sequence of square wave**

From the list, choose either high-low or low-high:



# Diamond Quartz-MM-10 Counter FM

---

- 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

# Diamond Quartz-MM-10 Counter FM & ARM

---

**Purpose** Quartz-MM-10 Counter FM & ARM block

**Library** xPC Target Library for Diamond

**Note** The Quartz-MM 10 has two AM9513A chips with five counters. 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.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Variable frequency: Double Arm: Double	<0.5 disarmed ≥0.5 armed

## Block Parameters

### Counter

From the list, choose 1, 2, 3, 4, 5, 6, 7, 8, 9, or 10 to select which counter is used with this driver block. In each case, one block is needed for each counter.

### Frequency base

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### Output duty cycle

Enter a value between 0 and 1 to set the duty cycle of the square wave. The duty cycle is held fixed during execution of the target application.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the **Output duty cycle** defined in the setting above defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running. If the application is running, the second input port controls whether the counter is armed or disarmed. This parameter sets the TTL level when the counter is disarmed.

## **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle).

## **Base address**

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

0x300

# Diamond Quartz-MM-10 PWM Capture

---

**Purpose** Quartz-MM-10 PWM Capture block

**Library** xPC Target Library for Diamond

**Note** 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. Leave both IN pins unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1&2, 2&3, 3&4, 4&5, 6&7, 7&8, 8&9, or 9&10. This selects which two counters the driver block uses to determine the PWM. Use one block for each pair of counters. No two blocks should use the same counter. For example, use counters 1&2 for one block and 3&4 for a second block. Do not use a combination like 1&2 and 2&3.

### **Frequency base**

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle)

# Diamond Quartz-MM-10 PWM Capture

---

## **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 Quartz-MM-10 FM Capture

---

**Purpose** Quartz-MM-10 FM Capture block

**Library** xPC Target Library for Diamond

**Note** This block programs the AM9513A for capturing FM signals.  
There is one output for relative frequency compared to the base frequency. To get the actual frequency, multiply the base frequency by the relative frequency.  
The FM signal must enter the pin of the corresponding counter channel named GATE. Leave the IN pin unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, 4, 5, 6, 7, 8, 9, or 10. This selects which counter the driver block uses to determine the FM. In each case, one block is needed for each counter.

### **Frequency base**

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (duty cycle).

### **Base address**

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

0x300

## Board

Diamond Quartz-MM-5

## General Description

The Quartz-MM 5 has 8 digital input lines, 8 digital output lines, and 10 counter/timers.

xPC Target supports this board with these driver blocks:

- Diamond Quartz-MM-5 Digital Input
- Diamond Quartz-MM-5 Digital Output
- Diamond Quartz-MM-5 Counter PWM
- Diamond Quartz-MM-5 Counter PWM & ARM
- Diamond Quartz-MM-5 Counter FM
- Diamond Quartz-MM-5 Counter FM & ARM
- Diamond Quartz-MM-5 PWM Capture
- Diamond Quartz-MM-5 FM Capture
- Diamond Quartz-MMxx

## Board Characteristics

Board name	Quartz-MM-5
Manufacturer	Diamond Systems Corporation
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	Yes — a channel can be accessed by at most one block
Multiple board support	Yes

# Diamond Quartz-MM-5 Digital Input

---

**Purpose** Quartz-MM-5 Digital Input block

**Library** xPC Target Library for Diamond

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

## **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 of 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



# Diamond Quartz-MM-5 Digital Output

**Purpose** Quartz-MM-5 Digital Output block

**Library** xPC Target Library for Diamond

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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

# Diamond Quartz-MM-5 Counter PWM

---

**Purpose** Quartz-MM-5 Counter PWM block

**Library** xPC Target Library for Diamond

**Note** The Quartz-MM5 has one AM9513A chip with 5 counters.

The Quartz-MM5 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 which defines the variable duty cycle between 0 and 1. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, or 4 to select which counter is used with this driver block. In each case, one block is needed for each counter.

### **Frequency base**

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### **Relative output frequency**

Enter a value between 0 and .25\***Frequency base**. The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 17.5 kHz, then choose F2=400kHz as the Frequency Base and enter 0.04375 as the Relative Output Frequency.  $400 \text{ kHz} \times 0.04375 = 17.5 \text{ kHz}$

# Diamond Quartz-MM-5 Counter PWM

---

---

**Note** Your control over the output frequency is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and gets 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

# Diamond Quartz-MM-5 Counter PWM & ARM

---

**Purpose** Quartz-MM-5 Counter PWM & ARM block

**Library** xPC Target Library for Diamond

**Note** The Quartz-MM5 has one AM9513A chip with 5 counters.

The Quartz-MM5 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 to arm and disarm the counter by the second block input. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Duty cycle: Double Arm: Double	0 to 1 <0.5 disarmed ≥0.5 armed

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, 4, or 5 to select which counter is used with this driver block. In each case, one block is needed for each counter.

### **Frequency base**

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### **Relative output frequency**

Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 17.5 kHz, then choose F2=400kHz as the Frequency Base and enter 0.04375 as the Relative Output Frequency.  $400 \text{ kHz} \times 0.04375 = 17.5 \text{ kHz}$

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running. If the application is running, the second input port controls whether the counter is armed or disarmed. This parameter sets the TTL level when the counter is disarmed.

## **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

## **Base address**

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

0x300

# Diamond Quartz-MM-5 Counter FM

---

**Purpose** Quartz-MM-5 Counter FM block

**Library** xPC Target Library for Diamond

**Note** The Quartz-MM5 has one AM9513A chip with 5 counters.

The Quartz-MM5 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 PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, 4, or 5 to select which counter is used with this driver block. In each case, one block is needed for each counter.

### **Frequency base**

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### **Output duty cycle**

Enter a value between 0 and 1 to set the duty cycle of the square wave. The Duty Cycle is held fixed during execution of the target application.

### **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.

# Diamond Quartz-MM-5 Counter FM

---

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 gets 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 corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Diamond Quartz-MM-5 Counter FM & ARM

---

**Purpose** Quartz-MM-5 Counter FM & ARM block

**Library** xPC Target Library for Diamond

**Note** The Quartz-MM5 has one AM9513A chip with 5 counters.

The Quartz-MM5 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 to arm and disarm the counter by 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	Variable frequency: Double Arm: Double	<0.5 disarmed ≥0.5 armed

## Block Parameters

### Counter

From the list, choose 1, 2, 3, 4, 5, 6, 7, 8, 9, or 10 to select which counter is used with this driver block. In each case, one block is needed for each counter.

### Frequency base

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### Output duty cycle

Enter a value between 0 and 1 to set the duty cycle of the square wave. The Duty Cycle is held fixed during execution of the target application.



# Diamond Quartz-MM-5 Counter FM & ARM

---

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

## **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

# Diamond Quartz-MM-5 PWM Capture

---

**Purpose** Quartz-MM-5 PWM Capture block

**Library** xPC Target Library for Diamond

**Note** This block programs the AMD9513A for capturing PWM signals by using two counters. One counter measures the cycle duration, and the other counter measures the duration 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 has to enter the pins named GATE of both corresponding counter channels (parallel wiring). Leave both IN pins unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1&2, 2&3, 3&4, or 4&5. This selects which two counters the driver block uses to determine the PWM. Use one block for each pair of counters. No two blocks should use the same counter. For example, use counters 1&2 for one block and 3&4 for a second block. Do not use a combination like 1&2 and 2&3.

### **Frequency base**

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

# Diamond Quartz-MM-5 PWM Capture

---

## **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 Quartz-MM-5 FM Capture

---

**Purpose** Quartz-MM-5 FM Capture block

**Library** xPC Target Library for Diamond

**Note** This block programs the AMD9513A 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 has to enter the pin named GATE of the corresponding counter channel. Leave the IN pin unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, 4 or 5. This selects which counter the driver block uses to determine the FM. In each case, one block is needed for each counter.

### **Frequency base**

From the list, choose F1=4MHz, F2=400kHz, F3=40kHz, F4=4kHz, or F5=400Hz to set the base frequency.

### **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

### **Base address**

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

0x300

**Purpose** Diamond Quartz-MMxx block

**Description** You can use this block to program the AMD9513A 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.

# Diamond Quartz-MMxx

---

**Purpose** Quartz-MMxx block

**Description** 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.

**Board** Diamond Ruby-MM

**General Description**

The Diamond Ruby-MM is a PC104 I/O board with 4 or 8 single analog output (D/A) channels (12-bit), unipolar and bipolar operation, +/- 10V, +/- 5V, 0-10V, 0-5V fixed ranges, +/- 2.5V, 0-2.5V user-adjustable ranges, 24 digital input and output lines.

xPC Target supports this board with these driver blocks:

- Diamond Ruby-MM Analog Output (D/A)
- Diamond Ruby-MM Digital Input
- Diamond Ruby-MM Digital Output

**Board Characteristics**

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

# Diamond Ruby-MM Analog Output (D/A)

---

**Purpose** Ruby-MM Analog Output block

**Library** xPC Target Library for Diamond

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

## **Block Parameters**

### **Channel vector**

List the output channels as a vector. Up to 8 different channels can be listed. To specify the first three channels, enter

[1,2,3]

The board comes in two different hardware versions. If the 4-channel version of the hardware is used and the channels 5-8 are listed, the those outputs will not show an error, but the data will be ignored.

### **Range**

The output range of the board is selected with jumpers on the board. Each group of 4 channels can be jumpered for any of the available ranges. The range you select in the Block Parameters must correspond to the range specified by the jumper settings or you will obtain incorrect results.

### **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.



# Diamond Ruby-MM Analog Output (D/A)

---

## **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 in hexadecimal (such as 0x300) as selected with the hardware jumpers on the board.

---

**Note** Please consult the appropriate Diamond Ruby-MM hardware manuals for more information on jumper settings and I/O connections.

---

# Diamond Ruby-MM Digital Input

---

**Purpose** Ruby-MM Digital Input block

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital input lines to be read. This driver allows the selection of up to 8 individual digital input lines in any order. The number of elements defines the number of digital lines used.

For example, to use all of the digital inputs enter

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

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

### **Port**

From the **Port** list, choose either A, B, or C. The **Port** parameter defines which port is used for this driver block. Each port has a maximum of 8 digital inputs. In each case, one block is needed for each port.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **Base address**

Enter the base address of the board in hexadecimal (such as 0x300) as selected with the hardware jumpers on the board.

# Diamond Ruby-MM Digital Input

---

---

**Note** Please consult the appropriate Diamond Ruby-MM hardware manuals for more information on jumper settings and I/O connections.

---

# Diamond Ruby-MM Digital Output

---

**Purpose** Ruby-MM Digital Output block

**Library** xPC Target Library for Diamond

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital output lines to drive. This driver allows the selection of up to 8 individual digital output lines in any order. The number of elements defines the number of digital lines used.

For example, to use all of the digital inputs enter

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

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

### **Port**

From the **Port** list, choose either A, B, or C. The **Port** parameter defines which port is used for this driver block. Each port has a maximum of 8 digital outputs. 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.

# Diamond Ruby-MM Digital Output

---

## **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 the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board in hexadecimal (such as 0x300) as selected with the hardware jumpers on the board.

---

**Note** Please consult the appropriate Diamond Ruby-MM hardware manuals for more information on jumper settings and I/O connections.

---

# Diamond Ruby-MM-416

---

**Board** Diamond Ruby-MM-416

**General Description** 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 Target supports this board with these driver blocks:

- Diamond Ruby-MM-416 Analog Output (D/A)
- Diamond Ruby-MM-416 Digital Input
- Diamond Ruby-MM-416 Digital Output

<b>Board Characteristics</b>	Board name	Ruby-MM-416
	Manufacturer	Diamond Systems Corporation
	Bus type	ISA (PCI/104)
	Access method	I/O Mapped
	Multiple block instance support	D/A:No DIO:Yes
	Multiple board support	Yes

# Diamond Ruby-MM-416 Analog Output (D/A)

**Purpose** Ruby-MM-416 Analog Output block

**Library** xPC Target Library for Diamond

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

## 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

<b>Input Range (V)</b>	<b>Range Code</b>
-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.

# Diamond Ruby-MM-416 Analog Output (D/A)

---

## **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



# Diamond Ruby-MM-416 Digital Input

**Purpose** Ruby-MM-416 Digital Input block

**Library** xPC Target Library for Diamond

**Note** 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

## 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.

# Diamond Ruby-MM-416 Digital Input

---

## **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

# Diamond Ruby-MM-416 Digital Output

**Purpose** Ruby-MM-416 Digital Output block

**Library** xPC Target Library for Diamond

**Note** 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

## 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.

# Diamond Ruby-MM-416 Digital Output

---

## **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

**Board** Diamond Ruby-MM-1612

**General Description**

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 Target supports this board with these driver blocks:

- Diamond Ruby-MM-1612 Analog Output (D/A)
- Diamond Ruby-MM-1612 Digital Input
- Diamond Ruby-MM-1612 Digital Output

**Board Characteristics**

Board name	Ruby-MM-1612
Manufacturer	Diamond Systems Corporation
Bus type	PC/104
Access method	I/O Mapped
Multiple block instance support	D/A:No DIO:Yes
Multiple board support	Yes

# Diamond Ruby-MM-1612 Analog Output (D/A)

---

**Purpose** Ruby-MM-1612 Analog Output block

**Library** xPC Target Library for Diamond

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

## **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 may 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 or 0 to +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. 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.

# Diamond Ruby-MM-1612 Analog Output (D/A)

<b>B1</b>	<b>B2</b>	<b>Configuration</b>
±10 V	±10 V	C11
0-10 V	±10 V	C41
±5 V	±5 V	C22
±2.5	±5 V	C32
0-5 V	±5 V	C52
0-2.5 V	±5 V	C62
±5 V	±2.5 V	C23
±2.5 V	±2.5 V	C33
0-5 V	±2.5 V	C53
0-2.5 V	±2.5 V	C63
±10 V	0-10 V	C14
0-10 V	0-10 V	C44
±5 V	0-5 V	C25
±2.5 V	0-5 V	C35
0-5 V	0-5 V	C55
0-2.5 V	0-5 V	C65
±5 V	0-2.5 V	C26
±2.5 V	0-2.5 V	C36
0-5 V	0-2.5 V	C56
0-2.5 V	0-2.5 V	C66

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

# Diamond Ruby-MM-1612 Analog Output (D/A)

---

results in an output voltage of  $(1.2 / 2.5) * 2.3 \text{ V} = 1.1 \text{ V}$ . See the board manual for details.

## **Reset vector**

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

## **Initial value vector**

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

## **Sample time**

Base sample time 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



# Diamond Ruby-MM-1612 Digital Input

**Purpose** Ruby-MM-1612 Digital Input block

**Library** xPC Target Library for Diamond

**Note** 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.

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

## 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.

# Diamond Ruby-MM-1612 Digital Input

---

## **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

# Diamond Ruby-MM-1612 Digital Output

**Purpose** Ruby-MM-1612 Digital Output block

**Library** xPC Target Library for Diamond

**Note** 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

## 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.

### 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

# Diamond Ruby-MM-1612 Digital Output

---

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**

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

# General Standards

---

This chapter describes General Standards I/O boards supported by xPC Target (<http://www.generalstandards.com>).

“Overview of PMC-ADADIO  
Functionality” on page 21-2

Describes of how the PMC-ADADIO blocks can be configured to interact with one another and other blocks.

General Standards PMC-ADADIO

High-performance multifunction board that has eight 16-bit analog-to-digital converters and eight 16-bit digital-to-analog outputs. The board also has 8 bits of TTL level digital I/O.

General Standards PMC-16AO-12

High-speed board that has 12 16-bit analog outputs.

“Boards and Blocks — Alphabetical  
List” on page 21-14

Description of block parameters for General Standards driver blocks.

## Overview of PMC-ADADIO Functionality

The PMC-ADADIO board is an analog I/O PCI mezzanine card (PMC) device that can be used for a number of applications, such as data acquisition and process monitoring.

xPC Target supports this board with A/D and D/A driver blocks. The following xPC Target driver blocks control the A/D functionality of the PMC-ADADIO board:

- General Standards PMC-ADADIO Analog Input (A/D) Start
- General Standards PMC-ADADIO Analog Input (A/D) Read

The following xPC Target driver blocks control the D/A functionality of the PMC-ADADIO board:

- General Standards PMC-ADADIO Analog Output Write Block
- General Standards PMC-ADADIO Analog Output (D/A) Update

The use of these drivers differ slightly from other boards. Of particular note are the Boolean enable ports (labelled E) that most of the A/D and D/A blocks have for input and/or output. These enable ports perform the following:

- Control block action. If the value of the input enable port is true, the block executes. If the value is false, the block does not execute. Most blocks also have output enable ports. The output enable port has the same value as the input enable port. This allows the control block action value to be passed to successive blocks.
- Ensure that A/D and/or D/A blocks execute in the correct order. For example, the A/D Start block starts the analog to digital conversion of the channels selected by the A/D Read block. This block must finish its operation before the A/D Read block can execute. If the A/D Read block executes first, the A/D Read block waits indefinitely for the A/D conversion to complete.

An input enable port can have an Enable Signal block connected to the port. The Enable Signal block generates an input enable signal for the A/D and D/A

blocks. If you do not connect an Enable Signal block to the A/D or D/A block, the input enable port has a constant value of 1, or 'true.'

A typical A/D block configuration for analog input operation connects the AD Start block and AD Read block. Because the A/D Start block can take several microseconds to perform the analog to digital conversion of the channels selected by the A/D Read block, you can perform other operations in the meantime. For example, you can insert a typical D/A block configuration between the AD Start and Read blocks. A typical configuration for analog output operation connects the DA Write and DA Update blocks.

This section describes how to use the PMC-Adadio blocks to create a model that interleaves the analog input and analog output operations. It has the following topics:

- "A/D Blocks" on page 21-3
- "Create Enable Signal Blocks" on page 21-5
- "D/A Blocks" on page 21-7
- "Interleaving Analog Input and Analog Output Blocks" on page 21-10
- "Using Multiple Boards for Simultaneous Analog to Digital Conversion" on page 21-12

## A/D Blocks

A typical A/D block configuration for analog input operation connects the AD Start block and AD Read block. The AD Start block converts the data of the channels selected by the AD Read block.

### Adding A/D Blocks to a Model for Analog Input

**1** In the MATLAB Command Window, type

```
xpcLib
```

The xPC Target driver block library opens.

**2** Double-click the A/D group block.

A window with blocks for A/D drivers opens.

- 3** Double-click the General Standards group block.

A window with blocks for General Standards opens.

- 4** From the File menu, click **New -> Model**.

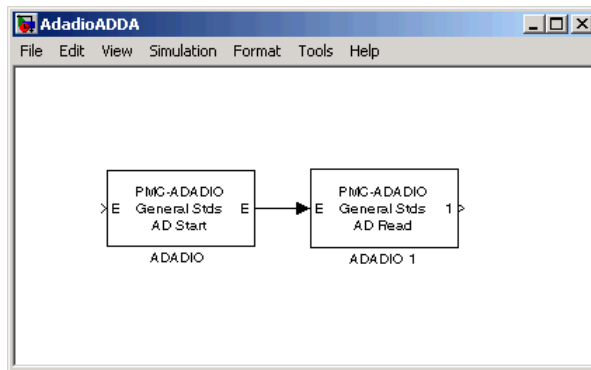
- 5** Drag and drop an AD Start block from the General Standards window to the new model.

By default, this block should have the **Enable input port** and **Enable output port** check boxes selected.

- 6** Drag and drop an AD Read block from the General Standards window to the new model.

By default, this block has the **Enable input port** and **Enable output port** check boxes selected. Double-click the AD Read block and clear the **Enable output port** check box. Clearing this check box prohibits the block from passing the Boolean value from the input enable port to the output enable port.

- 7** Connect the AD Start block to the AD Read block.



Note the following:

- No signal has been connected to the AD Start block's input enable port, E, so the port has a default value of 'true'. Accordingly, the output enable port of the AD Start block and input enable port of the AD Read block also have a value of 'true'. You can drag and drop a Ground block from



the Simulink Source library and connect that block to the unconnected input port of AD Start to prevent build errors until you add another block. You can drag and drop a Terminator block from the Simulink Sink library and connect that block to unconnected output ports of AD Read to prevent build errors until you add other blocks.

- Connecting the output enable port of the AD Start block to the input enable port of the AD Read block ensures that the AD Start block executes before the AD Read block. The AD Start block initiates the A/D conversion. The Read block waits until the conversion has completed before putting the results on its output port.

---

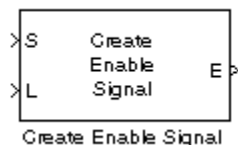
**Note** The Start block must execute before every call to the Read block. If the Read block is executed without the Start block, the system hangs because it is waiting for data to be available.

---

**8** From the File menu, select **Save As**. Browse to a writable directory and enter a unique model name, for example, AdadioADDA. Then click **Save**. Your next task is to add a Create Enable Signal block to this model. See “Adding Enable Signal Blocks to A/D Blocks” on page 21-6.

## Create Enable Signal Blocks

The A/D and D/A series of blocks both have Create Enable Signal blocks.



You can use a Create Enable Signal block to generate an input enable signal for A/D and D/A blocks. You can connect a signal generator to the input of the Create Enable Signal block to control the output enable port. You can then connect the output E port of the Create Enable Signal block to the input E port of an A/D or D/A block.

## Adding Enable Signal Blocks to A/D Blocks

This procedure assumes that you have a model named AdadioADDA (see “Adding A/D Blocks to a Model for Analog Input” on page 21-3). Add a Create Enable Signal block to generate an input enable signal for the AD Start block. If you have Ground or Terminator blocks, remove them as you make connections to the additional blocks.

- 1** If your model AdadioADDA is not already open, in the MATLAB Command Window, type

```
AdadioADDA
```

The model opens.

- 2** In the MATLAB window, type

```
xpcLib
```

The xPC Target library opens.

- 3** Double-click the A/D group block.

A window with blocks for A/D drivers opens.

- 4** Double-click the General Standards group block.

A window with blocks for General Standards opens.

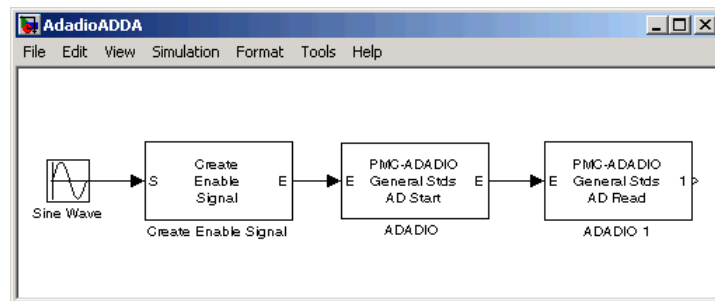
- 5** Drag and drop a Create Enable Signal block from the General Standards window to the new model.

- 6** Double click the Create Enable Signal block and deselect the Show input port for thresholding signal of type double.

The Create Enable block has an output enable port, E, that can provide an input enable signal for any of the other PMC-ADADIO blocks. The Boolean value of this output enable port is controlled by an input port, S, and (optionally) an input port, L. When the S port is connected to the output of an arbitrary block B, the Create Enable Signal block executes immediately after block B executes. The L port is a level-sensitive thresholding port that allows an attached signal to control the Boolean value at the output enable port E.

- 7** Click **Start > Simulink > Library Browser**.
  - 8** Click **Sources**. The Sources pane displays the included blocks.
  - 9** Drag and drop a Sine Wave block to the new model.
  - 10** Connect the output port of the Sine Wave block to the input port S of the Create Enable Signal block.
- Connecting the Sine Wave to the Create Enable Signal block triggers the sequence chain of the ADADIO blocks.
- 11** In the new model, connect the S port of the Create Enable Signal block to the Sine Wave block.
  - 12** In the new model, connect the AD Start block E port to the Create Enable Signal E port.

The output enable port, E, of the Create Enable Signal block provides the first Boolean output to feed into the other ADADIO driver blocks.



- 13** From the File menu, select **Save**.

Your next task is to set up the D/A blocks to provide the analog output for the analog input blocks.

## D/A Blocks

A typical D/A block configuration for analog output operation connects the DA Write block and DA Update block. The DA Update block converts the data that the DA Write block puts out.

## **Adding D/A Blocks to a Model for Analog Output**

- 1** If the model AdadioADDA is not already open, in the MATLAB Command Window, type

```
AdadioADDA
```

The model opens.

- 2** In the MATLAB window, type

```
xpcLib
```

The xPC Target library opens.

- 3** Double-click the D/A group block.

A window with blocks for D/A drivers opens.

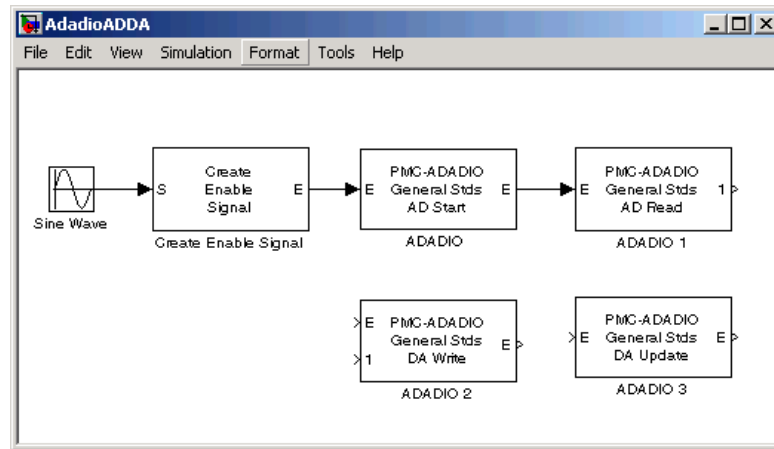
- 4** Double-click the General Standards group block.

A window with blocks for General Standards opens.

- 5** Drag and drop a DA Write block from the General Standards window to the new model.

- 6** Drag and drop a DA Update block from the General Standards window to the new model.

## 7 Connect the DA Update block to the DA Write block.



Note the following:

- No signal has been connected to the DA Write block's input enable port, E, so the port has a default value of 'true'. Accordingly, the output enable port of the DA Write block and input enable port of the DA Update block also have a value of 'true'.
- Connecting the output enable port of the DA Write block to the input enable port of the DA Update block ensures that the DA Write block executes before the DA Update block. The DA Write block loads the PMC-ADADIO registers in preparation for the D/A conversion. The DA Update block waits until the loading has completed before initiating the D/A conversion. As with the AD Start and Read blocks, the DA Write and Update blocks do not need to be directly connected to each other. You can interleave other blocks if you want.

You can interleave the analog output blocks between the AD Start and AD Read blocks. Such a configuration allows the analog output block to perform concurrently with the A/D conversion by the AD Start block. To connect to the AD Start and AD Read blocks, the interleaving block(s) must have an input enable port and an output enable port. See "Interleaving Analog Input and Analog Output Blocks" on page 21-10.

## Interleaving Analog Input and Analog Output Blocks

After the AD Start block executes, the acquisition hardware becomes busy with the operation. If you have the AD Read block execute immediately, it will idle waiting for the hardware to finish the acquisition. Rather than allowing idle cycles, you can insert other blocks between the AD Start and AD Read block. You can insert

- An atomic subsystem that has a pass through input for the enable signal from the AD Start to the AD Read blocks. This type of system enforces an execution order where the inserted subsystem executes between the other two blocks.
- A pair of blocks that already have an enable port, such as DA Write and DA Update. Because the DA Write and DA Update blocks already have an enable port, you do not have to include them in another subsystem to ensure correct execution order. See the following enable line for the execution order:
  - a** AD Start enable out to DA Write enable in
  - b** DA Write enable out to DA Update enable in
  - c** DA Update enable out to AD Read enable in

This results in the time sequence: **AD Start > DA Write > DA Update > AD Read**. By the time AD Read executes, the hardware has either finished or is much closer to finishing. Less time is wasted than if you use no interleaving. Note that the data input to DA Write comes from another part of the model. Typically, it will be the value calculated from the AD Read from the previous time step.

The following procedure assumes that you have a model named AdadioADDA that has AD Start and Read blocks, an Enable Create Signal block, and DA Write and Update blocks:

- 1** If the model AdadioADDA is not already open, in the MATLAB Command Window, type

```
AdadioADDA
```

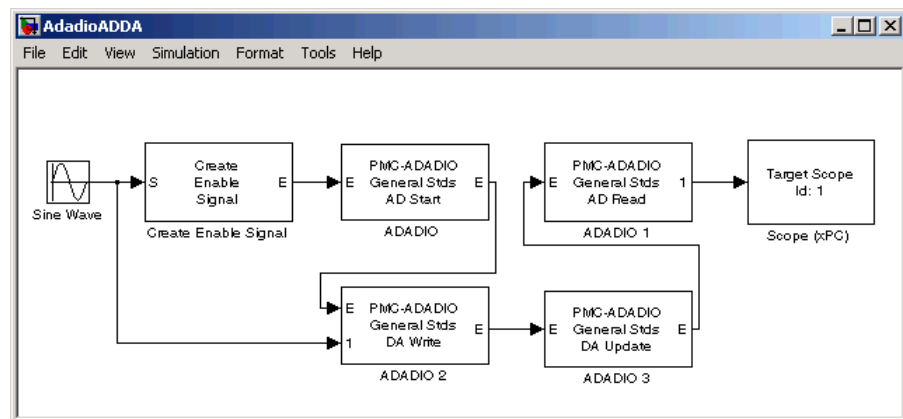
The model opens.

- 2** In the MATLAB window, type
 

```
xpclib
```

The xPC Target library opens.
- 3** Disconnect the AD Start block from the AD Read block.
 

Perform this step to insert the intermediate DA blocks between the AD Start and Read blocks.
- 4** Connect the output E port of the AD Start block to the input E port of the DA Write block.
- 5** Connect the output port of the Sine Wave block to the input 1 port of the DA Write block.
- 6** Connect the output E port of the DA Write block to the input E port of the DA Update block.
- 7** Connect the output E port of the DA Update block to the input E port of the AD Read block.
- 8** At the xPC Target library, double-click the Misc group block
- 9** Drag and drop the Scope (xPC) block into the AdadioADDA model.
- 10** Connect the output E port of the AD Read block to the Scope (xPC) block.



**11** From the File menu, select **Save**.

You can build and run the model and download it to your target PC like any other xPC Target model.

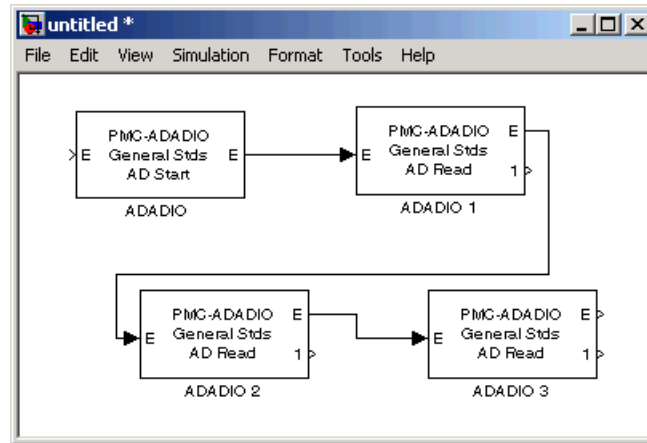
## **Using Multiple Boards for Simultaneous Analog to Digital Conversion**

The previous topics describe how to set up a model for up to eight channels performing simultaneous analog to digital conversion. You can increase the number of channels for this conversion by using two or more PMC-ADADIO boards, configured in a master/slave configuration.

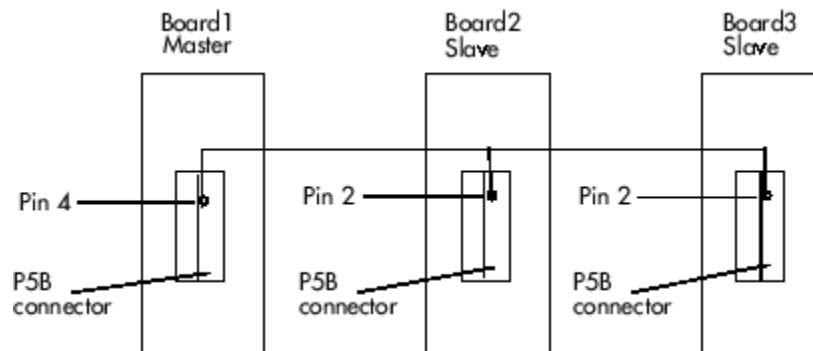
In such a configuration, observe the following guidelines:

- Decide how many slave PMC-ADADIO boards you want in your configuration.
- Identify one PMC-ADADIO as the master board.
- Create a model that uses the master PMC-ADADIO board. This model must contain an A/D Start block for the master board.
- For each PMC-ADADIO board in your system, including the master and all slave boards, the model must contain an associated A/D Read block.
- Connect the Enable output port (E) of the A/D Start block of the master board to the Enable input port (E) of each of the slave A/D Read blocks. You can do this in one of the following ways. Note that the order in which the A/D Read blocks are connected is irrelevant.
  - Directly connect the Enable output port (E) of the A/D Start block to the Enable input port (E) of each of the slave A/D Read blocks.
  - Directly connect the Enable output port (E) of the A/D Start block to the Enable input port (E) of the first slave A/D Read block, then create a daisy chain of slave Enable input to Enable output ports for the slave A/D Read blocks. For example





- Connect pin4 of the P5B connector (INPUT TRIGGER READY signal) of the master board to pin 2 of the P5B connector (INPUT TRIGGER signal) of all the slave boards. For example




---

**Note** The pulse of the master board INPUT TRIGGER READY output occurs 250 nanoseconds after the software trigger in the A/D Start block. This results in a 250 nanosecond delay, relative to the master board, for the slave boards, but should still be considered simultaneous in most situations.

---

## **Boards and Blocks – Alphabetical List**

## Board

General Standards PMC-ADADIO

## General Description

The PMC-ADADIO is a high performance multifunction board that has eight 16 bit analog to digital converters and four 16 bit digital to analog outputs. Up to eight input channels are sampled simultaneously from a single acquisition start. The board also has 8 bits of TTL level digital I/O. All 8 bits are either inputs or outputs.

You can purchase the PMC-ADADIO board in either a memory mapped or an I/O mapped configuration. The driver detects this configuration and adjusts for it. In addition, you order the board for a specific factory configured voltage range. Consult the board's documentation to determine the voltage range of your board.

xPC Target supports this board with these driver blocks:

- General Standards Create Enable Signal
- General Standards PMC-ADADIO Analog Input (A/D) Start
- General Standards PMC-ADADIO Analog Input (A/D) Read
- General Standards PMC-ADADIO Analog Output Write Block
- General Standards PMC-ADADIO Analog Output (D/A) Update
- General Standards PMC-ADADIO Digital Input
- General Standards PMC-ADADIO Digital Output

Note that you cannot use the PMC-ADADIO Digital Input and Digital Output driver blocks simultaneously on the same board. Trying to do so results in an error.

## Board Characteristics

Board name	PMC-ADADIO
Manufacturer	General Standards
Bus type	PCI
Access method	Memory mapped or I/O mapped

# General Standards PMC-ADADIO

---

Multiple block instance support	No
Multiple board support	Yes

# General Standards Create Enable Signal

---

<b>Purpose</b>	Create Enable Signal block
<b>Library</b>	xPC Target Library for General Standards
<b>Description</b>	Use the Create Enable Signal block to determine whether a series of blocks executes as well as to start the sequence of the blocks' execution. For example, connect another block in your model to the input port S to trigger the sequence chain of the ADADIO blocks. The output enable port, E, provides the first Boolean output to feed into the other ADADIO driver blocks.
<b>Scaling Input to Output</b>	The values passed to the enable output port of this block are Boolean values.
<b>Block Parameters</b>	<p><b>Show input port for thresholding signal of type double</b> Select this check box to display the level input port, L. Connect the signal whose level you want to compare to the threshold value to this port.</p> <p>If this check box is not selected, then the output enable port has a Boolean value of true. The output enable port provides a link for sequencing the execution of your blocks.</p> <p><b>Enable threshold</b> Enter a threshold value. If the block input signal connected to the level input port, L, is above this threshold, the output enable port has a Boolean value of true. If the input signal is below this threshold, the output enable port has a Boolean value of false.</p>

# General Standards PMC-ADADIO Analog Input (A/D) Start

---

<b>Purpose</b>	PMC-ADADIO Analog Input Start block
<b>Library</b>	xPC Target Library for General Standards
<b>Scaling Input to Output</b>	The values passed to the enable input port and enable output port of this block are Boolean values.
<b>Block Parameters</b>	<p><b>Enable input port</b> Select this check box to display the enable input port that controls the execution of the A/D conversion. If the enable input signal is true, the A/D conversion begins. If the enable input signal is false, then the A/D conversion is not started. If this check box is not selected, then the enable input signal is assumed to be true.</p> <p><b>Enable output port</b> Select this check box to pass the Boolean value from the input enable port to the output enable port. You can connect this port to the input enable port on another block to control other blocks in the model and specify execution ordering.</p> <p><b>Sample time</b> Enter the base sample time or a multiple of the base sample time.</p> <p><b>PCI Slot (-1:autosearch)</b> If only one board of this type is in the target PC, enter  -1  to automatically locate the board.  If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type  getxpcpci</p>

# General Standards PMC-ADADIO Analog Input (A/D) Read

**Purpose** PMC-ADADIO Analog Input Read block

**Library** xPC Target Library for General Standards

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

**Note** The values passed to the enable input port and enable output port of this block are Boolean values.

## **Block Parameters**

### **Number of channels**

Enter a scalar between 1 and 8 to select the number of channels from which to acquire data. This board always acquires data from the channels in chronological order with no random access to channel selection. For example, if you enter 3, data is acquired from channels 1, 2 and 3. Select a number between 1 and 8 even though the hardware manual numbers the channels from 0 to 7.

### **Range option**

From the list, select +-10V, +-5V or +-2.5V as the input voltage range of the board. You must select the range that corresponds to the factory configured input range. Consult the board's documentation to determine the voltage range of your board. xPC Target uses this parameter to convert the data to floating point numbers.

### **Input coupling**

From the list, select Single-ended (8 channels) or Differential (8 channels). See the board's hardware manual for information on how to wire the board for these configurations.

### **Autocalibration**

The PMC-ADADIO can check its own calibration (autocalibration) against an internal voltage reference. Choose this check box to execute the autocalibration cycle when the model is downloaded to the xPC Target target PC. The calibration cycle takes several

# General Standards PMC-ADADIO Analog Input (A/D) Read

---

seconds to run. The output ports show a square wave during the cycle. Unless you subject this board to harsh conditions, you should only need to occasionally run autocalibration, for example, once a month.

Refer to the PMC-ADADIO documentation for further information on autocalibration.

## **Enable input port**

Select this check box to display an input port that controls writing data from the A/D ports on the hardware. If the enable input signal is true, then the output of the A/D converter is read and output to the block output port. If the enable input signal is false, then zero data is output to the block output port. If this check box is not selected, then the enable input signal is assumed to be true.

## **Enable output port**

Select this check box to pass the Boolean value from the input enable port to the output enable port. You can connect this port to the input enable port on another block to control other blocks in the model and specify execution ordering.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# General Standards PMC-ADADIO Analog Output Write Block

**Purpose** PMC-ADADIO Analog Output Write block

**Library** xPC Target Library for General Standards

**Note** Data from 1 to 4 input data streams is written to the hardware digital to analog output registers when the enable port also receives a true value. If the enable port is not used, then it is assumed to be true always. The output registers are held and not written through to the output converters until the analog output update block is executed.

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

The values passed to the enable input port and enable output port of this block are Boolean values.

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers between 1 and 4 to select the analog output lines to drive. Number the lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### **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. For example, if **Channel vector** is [ 2 3 ] and the **Reset vector** is [1], the action taken will be the same as if **Reset vector** was set to [1 1]. Both channels will be reset to their initial values when model execution is stopped.

# General Standards PMC-ADADIO Analog Output Write Block

---

## **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. When model execution is stopped, the corresponding position in **Reset vector** is checked. Depending on that value, the channel is either reset to the initial value or remains at the last value attained while the model was running. For example, assume that **Channel vector** is [2 5 8], **Reset vector** is [1 0 1], and **Initial value vector** is [2.3 5.6 0]. On initial download, channel 2 is set to 2.3 volts, channel 5 to 5.6 volts, and channel 8 to 0.0 volts. When the model is stopped, channel 2 resets to 2.3 volts, channel 5 remains at the last value attained, and channel 8 resets to 0.0 volts.

## **Range vector**

From the list, select +- 10V, +-5V or +-2.5V as the output voltage range of the board. You must select the range that corresponds to the factory configured output range. Consult the board's documentation to determine the voltage range of your board. xPC Target uses this parameter to convert the data from floating point numbers to integers.

## **Autocalibration**

The PMC-ADADIO can check its own calibration (autocalibration) against an internal voltage reference. Choose this check box to execute the autocalibration cycle when the model is downloaded to the xPC Target target PC. The calibration cycle takes several seconds to run. The output ports show a square wave during the cycle. Unless you subject this board to harsh conditions, you should only need to occasionally run autocalibration, for example, once a month.

Refer to the PMC-ADADIO documentation for further information on autocalibration.

# General Standards PMC-ADADIO Analog Output Write Block

---

## **Enable input port**

Select this check box to display an input enable port that controls writing data to the D/A ports on the hardware. If the enable input signal is true, then the block input data is written to the D/A ports on the hardware. If the enable input signal is false, then data is not written to the D/A ports on the hardware and the output remains at the previous value. If this check box is not selected, then the enable input signal is assumed to be true.

## **Enable output port**

Select this check box to pass the Boolean value from the input enable port to the output enable port. You can connect this port to the input enable port on another block to control other blocks in the model and specify execution ordering.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# General Standards PMC-ADADIO Analog Output (D/A) Update

---

<b>Purpose</b>	PMC-ADADIO Analog Output Update block
<b>Library</b>	xPC Target Library for General Standards
<b>Scaling Input to Output</b>	The values passed to the enable input port and enable output port of this block are Boolean values.
<b>Block Parameters</b>	<p><b>Enable input port</b> Select this check box to display an input enable port that controls writing data to the D/A ports on the hardware. If the enable input signal is true, then the block input data is written to the D/A ports on the hardware. If the enable input signal is false, then data is not written to the D/A ports on the hardware and the output remains at the previous value. If this check box is not selected, then the enable input signal is assumed to be true.</p> <p><b>Enable output port</b> Select this check box to pass the Boolean value from the input enable port to the output enable port. You can connect this port to the input enable port on another block to control other blocks in the model and specify execution ordering.</p> <p><b>Sample time</b> Enter the base sample time or a multiple of the base sample time.</p> <p><b>PCI Slot (-1:autosearch)</b> If only one board of this type is in the target PC, enter  -1  to automatically locate the board.</p>

# General Standards PMC-ADADIO Analog Output (D/A) Update

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# General Standards PMC-ADADIO Digital Input

---

**Purpose** PMC-ADADIO Digital Input block

**Library** xPC Target Library for General Standards

**Note** Note that you cannot use the PMC-ADADIO Digital Input and Digital Output driver blocks simultaneously on the same board. Doing so causes an error at model update or build.

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

The values passed to the enable input port and enable output port of this block are Boolean values.

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital input lines to be read. 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 of the digital inputs, enter

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

Number the lines beginning with 1 even though the board manufacturer starts numbering the lines with 0. Note that the 8 digital lines are either all input or all output.

### **Enable input port**

Select this check box to display an input enable port that controls writing data from the digital input ports on the hardware. If the enable input signal is true, then the digital input values are read and output to the block output port. If the enable input signal is

# General Standards PMC-ADADIO Digital Input

---

false, then zero data is output to the block output port. If this check box is not selected, then the enable input signal is assumed to be true.

## **Enable output port**

Select this check box to pass the Boolean value from the input enable port to the output enable port. You can connect this port to the input enable port on another block to control other blocks in the model and specify execution ordering.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# General Standards PMC-ADADIO Digital Output

---

**Purpose** PMC-ADADIO Digital Output block

**Library** xPC Target Library for General Standards

**Note** Note that you cannot use the PMC-ADADIO Digital Input and Digital Output driver blocks simultaneously on the same board. Doing so causes an error at model update or build.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

The values passed to the enable input port and enable output port of this block are Boolean values.

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital output lines to be written. 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 of the digital outputs, enter

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

Number the lines beginning with 1 even though the board manufacturer starts numbering the lines with 0. Note that the 8 digital lines are either all input or all output.

### **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. For example, if **Channel vector** is [ 2 3 ] and the **Reset vector** is [1], the action taken will be the same as if **Reset vector** was set to [1 1]. Both channels will be reset to their initial values when model execution is stopped.

## **Initial value vector**

The initial value vector contains the initial logical values for the output channels. Enter a scalar or a vector that is the same length as the channel vector. These values can only contain 1's and 0's. 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. When model execution is stopped, the corresponding position in **Reset vector** is checked. Depending on that value, the channel is either reset to the initial value or remains at the last value attained while the model was running. For example, assume that **Channel vector** is [2 5 8], **Reset vector** is [1 0 1], and **Initial value vector** is [1 1 0]. On initial download, channel 2 is set to high, channel 5 to high, and channel 8 to low. When the model is stopped, channel 2 resets to high, channel 5 remains at the last value attained, and channel 8 resets to low.

## **Enable input port**

Select this check box to display an input enable port that controls writing data to the digital output ports on the hardware. If the enable input signal is true, then the block input data is written to the digital output ports on the hardware. If the enable input signal is false, then data is not written to the digital output ports on the hardware and the output remains at the previous value. If this check box is not selected, then the enable input signal is assumed to be true.

# General Standards PMC-ADADIO Digital Output

---

## **Enable output port**

Select this check box to pass the Boolean value from the input enable port to the output enable port. You can connect this port to the input enable port on another block to control other blocks in the model and specify execution ordering.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** General Standards PMC-16AO-12

**General Description** The PMC-16AO12 board is a high speed board that has twelve 16-bit analog outputs. In this section, PMC-16AO12 refers to the board, PMC-16AO-12 refers to the xPC Target block.

xPC Target supports this board with these driver blocks:

- General Standards PMC-16AO-12 Analog Output

<b>Board Characteristics</b>	Board name	PMC-16AO12
	Manufacturer	General Standards
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	No
	Multiple board support	Yes

# General Standards PMC-16AO-12 Analog Output

---

**Purpose** PMC-16AO-12 Analog Output block

**Library** xPC Target Library for General Standards

**Scaling Input to Output** The analog output range of this board depends on the output range that was chosen at the time of purchase. This range can be  $+ -2.5$ ,  $+ -5.0$ , or  $+ -10.0$ .

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 12. This is a vector parameter that specifies the hardware output channels. For example, to produce output on channels 2 and 3, enter

[2 3]

All unspecified channels are set to 0.0 volts on output.

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

### 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. For example, if **Channel vector** is [ 2 3 ] and the **Reset vector** is [1], the action taken will be the same as if **Reset vector** was set to [1 1]. Both channels will be reset to their initial values when model execution is stopped.

## 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. When model execution is stopped, the corresponding position in **Reset vector** is checked. Depending on that value, the channel is either reset to the initial value or remains at the last value attained while the model was running. For example, assume that **Channel vector** is [2 5 8], **Reset vector** is [1 0 1], and **Initial value vector** is [2.3 5.6 0]. On initial download, channel 2 is set to 2.3 volts, channel 5 to 5.6 volts, and channel 8 to 0.0 volts. When the model is stopped, channel 2 resets to 2.3 volts, channel 5 remains at the last value attained, and channel 8 resets to 0.0 volts.

## Range

From the list, choose either +-2.5, +-5.0, or +-10.0. The PMC-16AO-12 is specified at purchase time as having one of these three ranges. Choose the range to match that for which the board is configured.

Note that the range you set does not alter any configuration on the board. It is used to scale the signal when it is converted to a 16-bit value to be written to the output D/A converter (DAC).

## Autocalibration

The PMC-16AO-12 can check its own calibration (autocalibration) against an internal voltage reference. Choose this check box to execute the autocalibration cycle when the model is downloaded to the xPC Target target PC. The calibration cycle takes several seconds to run. The output ports show a square wave during the cycle. The calibration cycle yields adjustment factors that are retained in onboard NVRAM. The PMC-16AO-12 reads these adjustment factors when the board is initialized. Unless you subject this board to harsh conditions, you should only need to occasionally run autocalibration, for example, once a month.

# General Standards PMC-16AO-12 Analog Output

---

Refer to the PMC-16AO-12 documentation for further information on autocalibration.

## **Ground sense**

Choose this check box to enable the hardware provided compensation for ground potential differences. Refer to the PMC-16AO-12 documentation for further information on ground sense.

## **Sampletime**

Enter the base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Humusoft

---

This chapter describes the Humusoft I/O board supported by xPC Target (<http://www.humusoft.cz/datacq/index.htm>).

Humusoft AD 512

I/O board with 8 single analog input (A/D) channels, 2 analog output (D/A) channels, 8 digital inputs, and 8 digital outputs.

## **Boards and Blocks**



**Board** Humusoft AD 512

**General Description**

The AD 512 is an I/O board with 8 single analog input (A/D) channels (12-bit) with a maximum sample rate of 100 kHz, 2 analog output (D/A) channels (12-bit), 8 digital inputs, and 8 digital outputs.

xPC Target supports this board with these driver blocks:

- Humusoft AD 512 Analog Input (A/D)
- Humusoft AD 512 Analog Output (D/A)
- Humusoft AD 512 Digital Input
- Humusoft AD 512 Digital Output

**Board Characteristics**

Board name	AD 512
Manufacturer	Humusoft
Bus type	ISA
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# Humusoft AD 512 Analog Input (A/D)

---

**Purpose** AD 512 Analog Input block

**Library** xPC Target Library for Humusoft

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8. This driver allows the selection of individual channels in any order. The number of elements defines the number of A/D channels used.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels entered in the channel vector. The range vector must be the same length as the channel vector. This driver allows a different range for each channel.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Humusoft AD 512 Analog Input (A/D)

---

For example, if the first channel is -10 to +10 volts and the second and fifth channels are 0 to +5 volts, enter

[ -10,5,5]

## **Sample time**

Model 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 jumper settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Humusoft AD 512 Analog Output (D/A)

---

**Purpose** AD 512 Analog Output block

**Library** xPC Target Library for Humusoft

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

**Block Parameters**

**Channel vector**

This parameter is a combined channel vector and range vector. The number of elements defines the number of D/A channels used.

Enter a range code for each of the channels used. This driver allows a different range for each channel with a maximum of two channels.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

**Sample time**

Enter the base sample time or a multiple of the base sample time.

# Humusoft AD 512 Analog Output (D/A)

---

## **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

# Humusoft AD 512 Digital Input

---

**Purpose** AD 512 Digital Input block

**Library** xPC Target Library for Humusoft

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

## **Channel vector**

Enter a numbers between 1 and 8. This driver allows the selection of individual digital line numbers in any order. The number of elements defines the number of digital input lines used.

For example, to use the first, second and fifth digital input lines, enter

[1,2,5]

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

## **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

# Humusoft AD 512 Digital Output

**Purpose** AD 512 Digital Output block

**Library** xPC Target Library for Humusoft

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Channel vector**

Enter a numbers between 1 and 8. This driver allows the selection of individual digital line numbers in any order. The number of elements defines the number of digital output lines used.

For example, to use the first, second and fifth digital output lines, enter

[1,2,5]

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

## **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





# Keithley

---

This chapter describes I/O boards supported by xPC Target (<http://www.keithley.com>).

Keithley DAS-1800HR	I/O board with 16 single or 8 differential analog input (A/D) channels, 4 digital input lines and 4 digital output lines.
Keithley KCPI-1801HC	I/O board with 64 single or 32 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 4 digital input and output lines.
Keithley KPCI-1802HC	I/O board with 64 single or 32 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 4 digital input and output lines.
“Boards and Blocks — Alphabetical List” on page 23-2	Description of block parameters for Keithley driver blocks.

## **Boards and Blocks – Alphabetical List**

**Board** Keithley DAS-1800HR

**General Description**

The DAS-1800HR is an I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 100 kHz, 4 digital input lines and 4 digital output lines.

xPC Target supports this board with these driver blocks:

- Keithley DAS-1800HR Analog Input (A/D)
- Keithley DAS-1800HR Digital Input
- Keithley DAS-1800HR Digital Output

**Board Characteristics**

Board name	DAS-1800HR
Manufacturer	Keithley
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Keithley DAS-1800HR Analog Input (A/D)

---

**Purpose** DAS-1800HR Analog Input block

**Library** xPC Target Library for Keithley Metrabyte

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

## **Block Parameters**

### **Channel vector**

If you select 16 channels (**Input coupling** parameter set to Single-ended channels or Single-ended common mode) enter numbers between 1 and 16 to select the number of individual channels used. If you select eight channels (**Input coupling** parameter set to Differential) enter numbers between 1 and 8 to select the A/D channels used. This driver allows the selection of individual A/D channels in any order. The number of elements defines the number of A/D channels used.

For example, to use the first, second and fifth channels, enter

[1,2,5]

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

### **Gain vector (1,2,4,8)**

Enter 1, 2, 4, or 8 for each of the channels in the channel vector to choose the gain code of that channel. The gain vector must be the same length as the channel vector. This driver allows the gain of each channel to be different.

Notice that by increasing the gain code the voltage range is decreased. The gain divides the input voltage range.

# Keithley DAS-1800HR Analog Input (A/D)

For example, if the first channel has a gain code of 1 (10 volt range) and the second and fifth channels have a gain code of 2 (5 volt range), enter

[1,2,2]

## Range vector

From the list, choose either Bipolar or Unipolar.

The range setting defines if the board is working in bipolar or unipolar input mode. This setting is the same for all of the selected channels.

The following table is a list of the ranges for this driver given the gain entered and the range chosen.

Gain	Bipolar Range (V)	Unipolar Range (V)
1	-10 to +10	0 to 10
2	-5 to + 5	0 to +5
4	-2.5 to 2.5	0 to 2.5
8	-1.25 to +1.25	0 to 1.25

## Input coupling

From the list, select one from the following list of input modes:

- Differential (8 channels)
- Single-ended (16 channels)
- Single-ended common mode (16 channels)

This choice must correspond to the MUX-switch setting on the board.

Common-mode is similar to single-ended mode but the negative wire of the source to be measured is connected to input AI-SENSE instead of LLGND.

# Keithley DAS-1800HR Analog Input (A/D)

---

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Keithley DAS-1800HR Digital Input

---

**Purpose** DAS-1800HR Digital Input block

**Library** xPC Target Library for Keithley Metrabyte

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

**Block Parameters**

**Number of channels**  
Enter a number between 1 and 8 to select the number of digital input lines used with this 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

# Keithley DAS-1800HR Digital Output

---

**Purpose** DAS-1800HR Digital Output block

**Library** xPC Target Library for Keithley Metrabyte

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Number of channels**

Enter a number between 1 and 4 to select the number of digital output lines used with this 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



**Board** Keithley KCPI-1801HC

**General Description**

The KCPI-1801 is an I/O board with 64 single or 32 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 333 kHz, 2 analog output (D/A) channels (12-bit), and 4 digital input and output lines.

xPC Target supports this board with these driver blocks:

- Keithley KPCI-1801HC Analog Input (A/D)
- Keithley KPCI-1801HC Analog Output (D/A)
- Keithley KPCI-1801HC Digital Input
- Keithley KPCI-1801HC Digital Output

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

**Board Characteristics**

Board name	KCPI-1801HC
Manufacturer	Keithley Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	A/D:No, D/A:Yes, Digital I/O:Yes
Multiple board support	Yes

# Keithley KPCI-1801HC Analog Input (A/D)

**Purpose** KPCI-1801HC Analog Input block

**Library** xPC Target Library for Keithley Metrabyte

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 64. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-5 to +5	-5
-1 to +1	-1
-0.1 to +0.1	-0.1
-0.02 to +0.02	-0.02
0 to +5	5

# Keithley KPCI-1801HC Analog Input (A/D)

Input Range (V)	Range Code
0 to +1	1
0 to +0.1	0.1
0 to +0.02	0.02

For example, if the first channel is -5 to +5 volts and the second and fifth channels are 0 to +1 volts, enter

[ -5, 1, 1 ]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

Coupling	Coupling Code	Description
single-ended	0	Analog input line connected to the positive input. Analog input ground (IGND) internally connected to the negative input.
differential	1	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

# Keithley KPCI-1801HC Analog Input (A/D)

---

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

```
[0,0,1]
```

The driver selects a second differential input 32 channels higher than the first channel. In the example above, the driver would select the 37th channel as a differential input with the fifth channel.

## **Sample time**

Model base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Keithley KPCI-1801HC Analog Output (D/A)

**Purpose** KPCI-1801HC Analog Output block

**Library** xPC Target Library for Keithley Metrabyte

**Note** The analog output range of this board is set -10 to +10 volts.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### 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

# Keithley KPCI-1801HC Analog Output (D/A)

---

the initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Base sample time of a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Keithley KPCI-1801HC Digital Input

**Purpose** KPCI-1801HC Digital Input block

**Library** xPC Target Library for Keithley Metrabyte

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 4 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 of the digital inputs for one port, enter

[ 1,2,3,4]

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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# Keithley KPCI-1801HC Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# Keithley KPCI-1801HC Digital Output

**Purpose** KPCI-1801HC Digital Output block

**Library** xPC Target Library for Keithley Metrabyte

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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.

# Keithley KPCI-1801HC Digital Output

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** Keithley KPCI-1802HC

**General Description**

The KPCI-1802 is an I/O board with 64 single or 32 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 333 kHz, 2 analog output (D/A) channels (12-bit), and 4 digital input and output lines.

xPC Target supports this board with these driver blocks:

- Keithley KPCI-1802HC Analog Input (A/D)
- Keithley KPCI-1802HC Analog Output (D/A)
- Keithley KPCI-1802HC Digital Input
- Keithley KPCI-1802HC Digital Output

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

**Board Characteristics**

Board name	KPCI-1802HC
Manufacturer	Keithley Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	A/D:No, D/A:Yes, Digital I/O:Yes
Multiple board support	Yes

# Keithley KPCI-1802HC Analog Input (A/D)

**Purpose** KPCI-1802HC Analog Input block

**Library** xPC Target Library for Keithley Metrabyte

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 64. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.25 to +1.25	-1.25
0 to +10	10

# Keithley KPCI-1802HC Analog Input (A/D)

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5
0 to +1.25	1.25

For example, if the first channel is -10 to +10 volts and the second and fifth channels are 0 to +5 volts, enter

[ -10,5,5]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

Coupling	Coupling Code	Description
Single- ended	0	Analog input line connected to the positive input. Analog input ground (IGND) internally connected to the negative input.
Differential	1	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

# Keithley KPCI-1802HC Analog Input (A/D)

---

[0,0,1]

The driver selects a second differential input 32 channels higher than the first channel. In the example above, the driver would select the 37th channel as a differential input with the fifth channel.

## **Sample time**

Model base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Keithley KPCI-1802HC Analog Output (D/A)

**Purpose** KPCI-1802HC Analog Output block

**Library** xPC Target Library for Keithley Metrabyte

**Note** The analog output range of this board is set -10 to +10 volts.

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

## **Block Parameters**

### **Channel vector**

Enter numbers 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### **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

# Keithley KPCI-1802HC Analog Output (D/A)

---

the initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Base sample time of a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# Keithley KPCI-1802HC Digital Input

**Purpose** KPCI-1802HC Digital Input block

**Library** xPC Target Library for Keithley Metrabyte

## Scaling Input to Output

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 4 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 of the digital inputs for one port, enter

[1,2,3,4]

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.

# Keithley KPCI-1802HC Digital Input

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Keithley KPCI-1802HC Digital Output

**Purpose** KPCI-1802HC Digital Output block

**Library** xPC Target Library for Keithley Metrabyte

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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

# Keithley KPCI-1802HC Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing (Formerly Computer Boards)

---

This chapter describes Measurement Computing I/O boards supported by xPC Target (<http://www.measurementcomputing.com>)

Measurement Computing CIO-CTR05	I/O board with 5 counter/timer channels.
Measurement Computing CIO-CTR10	I/O board with 10 counter/timer channels
Measurement Computing CIO-DAC08 (/12)	I/O board with 8 analog output (D/A) channels.
Measurement Computing CIO-DAC08/16	I/O board with 8 analog output (D/A) channels
Measurement Computing CIO-DAC16 (/12)	I/O board with 16 analog output (D/A) channels.
Measurement Computing CIO-DAC16/16	I/O board with 16 analog output (D/A) channels.
Measurement Computing CIO-DAS16/330	I/O board with 16 single or 8 differential analog input (A/D) channels, 4 digital input lines, and 4 digital output lines.
Measurement Computing CIO-DAS16/JR (/12)	I/O board with 16 single or 8 differential analog input (A/D) channels, 4 digital input lines, 4 digital output lines, and 3 counter/timers (16-bit).

Measurement Computing CIO-DAS16JR/16	I/O board with 16 single or 8 differential analog input (A/D) channels, 4 digital input lines, 4 digital output lines, and 3 counter/timers.
Measurement Computing CIO-DAS1601/12	I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, 32 digital input and output lines, and 3 counters.
Measurement Computing CIO-DAS1602/12	I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, 32 digital input and output lines, and 3 counters.
Measurement Computing CIO-DAS1602/16	I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, 32 digital I/O lines, and 3 counters
Measurement Computing CIO-DDA06 (/12)	I/O board with 6 analog output (D/A) channels, and 24 digital I/O lines.
Measurement Computing CIO-DDA06/16	I/O board with 6 analog output (D/A) channels, and 24 digital I/O lines.
Measurement Computing CIO-DIO24	I/O board with 24 digital I/O lines.
Measurement Computing CIO-DIO24H	I/O board with 24 digital I/O lines.
Measurement Computing CIO-DIO48	I/O board with 48 digital I/O lines.
Measurement Computing CIO-DIO48H	I/O board with 48 digital I/O lines.
Measurement Computing CIO-DIO96	I/O board with 96 digital I/O lines.
Measurement Computing CIO-DIO192	I/O board with 192 digital I/O lines.
Measurement Computing CIO-DO24DD	I/O board with 24 open-collector digital output lines.

Measurement Computing CIO-PDISO16	I/O board with 16 isolated digital input lines and 16 relay driven digital output lines.
Measurement Computing CIO-QUAD02	24-bit counting board with 2 channels. This board typically connects to incremental encoders.
Measurement Computing CIO-QUAD04	24-bit counting board with 4 channels. This board typically connects to incremental encoders.
Measurement Computing PC104-DAC06 (/12)	I/O board with 6 analog output (D/A) channels
Measurement Computing PC104-DAS16JR/12	I/O board with 16 single or 8 differential analog input channels, 4 digital input lines, and 4 digital output lines.
Measurement Computing PC104-DAS16JR/16	I/O board with 16 single or 8 differential analog input (A/D) channels, 4 digital input lines, and 4 digital output lines.
Measurement Computing CIO-DIO48	I/O board with 48 digital I/O lines
Measurement Computing PCI-CTR05	I/O board with 5 counter/timer channels.
Measurement Computing PCI-DAS1200	I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 24 digital input and output lines.
Measurement Computing PCI-DAS1200/JR	I/O board with 16 single or 8 differential analog input (A/D) channels, and 24 digital I/O lines.
Measurement Computing PCI-DAS1602/12	I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 24 digital input and output lines, and 3 counters.

Measurement Computing PCI-DAS1602/16	I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 24 digital input and output lines, and 3 counters.
Measurement Computing PCI-DDA02/12	I/O board with 2 analog output (D/A) channels, and 48 digital I/O lines.
Measurement Computing PCI-DDA02/16	I/O board with 2 analog output (D/A) channels, and 48 digital I/O lines.
Measurement Computing PCI-DDA04/12	I/O board with 4 analog output (D/A) channels, and 48 digital I/O lines
Measurement Computing PCI-DDA04/16	I/O board with 4 analog output (D/A) channels, and 48 digital I/O lines.
Measurement Computing PCI-DDA08/12	I/O board with 8 analog output (A/D) channels, and 48 digital I/O lines.
Measurement Computing PCI-DDA08/16	I/O board with 8 analog output (A/D) channels, and 48 digital I/O lines.
Measurement Computing CIO-DIO24	I/O board with 24 digital I/O lines
Measurement Computing CIO-DIO24H	I/O board with 24 digital I/O lines.
Measurement Computing CIO-DIO48H	I/O board with 48 digital I/O lines.
Measurement Computing PCI-DIO96H	I/O board with 96 digital I/O lines.
Measurement Computing PCI-DIO96	I/O board with 96 digital I/O lines.
Measurement Computing PCI-PDIS08	I/O board with 8 inputs and 8 relay outputs.
Measurement Computing PCI-PDIS016	I/O board with 16 inputs and 16 relay outputs.



---

Measurement Computing  
PCIM-DAS1602/16

I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, and 24 digital input and output lines, and 3 counters.

Measurement Computing  
PCIM-DDA06/16

I/O board with 6 analog output (D/A) channels, and 24 digital I/O lines.

Measurement Computing  
PCI-DUAL-AC5

I/O board with 48 digital I/O lines.

Measurement Computing  
PCI-QUAD04

24-bit counting board with 4 channels. This board typically connects to incremental encoders.

Measurement Computing  
PCI-DAS-TC

I/O board with 16 differential analog thermocouple input channels. The thermocouple signals are converted by a high-frequency synchronous V-F A/D converter.

“Boards and Blocks —  
Alphabetical List” on page  
24-6

Description of block parameters for  
Measurement Computing driver blocks.

## **Boards and Blocks – Alphabetical List**

# Measurement Computing CIO-CTR05

---

## Board

Measurement Computing CIO-CTR05

## General Description

The CIO-CTR05 is an I/O board with five counter/timer channels (16-bit).

It contains one AM9513A counter/timer chip. For additional information about the various counter/timer modes of that chip see the AM9513A data sheet which is part of the board documentation.

xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-CTR05 Counter PWM
- Measurement Computing CIO-CTR05 Counter PWM & ARM
- Measurement Computing CIO-CTR05 Counter FM
- Measurement Computing CIO-CTR05 Counter FM & ARM
- Measurement Computing CIO-CTR05 PWM Capture
- Measurement Computing CIO-CTR05 Frequency Capture
- Measurement Computing CIO-CTRxx

## Board Characteristics

Board name	CIO-CTR05
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing CIO-CTR05 Counter PWM

---

**Purpose** CIO-CTR05 Counter PWM block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-CTR05 has one AM9513A chip with five counters.  
The CIO-CTR05 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 which defines the variable duty cycle between 0 and 1. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency. The XTAL frequency is assumed to be 1 MHz, therefore the jumper on the CIO-CTR05 has to be in position 1 MHz not 5 MHz.

### **Relative output frequency**

Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 17.5 kHz, then choose F2=100kHz as the **Frequency base** and enter 0.175 as the **Relative output frequency**.  $100 \text{ kHz} \times 0.175 = 17.5 \text{ kHz}$

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and gets 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

# Measurement Computing CIO-CTR05 Counter PWM & ARM

---

**Purpose** CIO-CTR05 Counter PWM & ARM block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-CTR05 has one AM9513A chip with five counters.

The CIO-CTR05 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 to arm and disarm the counter by the second block input. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Duty cycle: Double Arm: Double	0 to 1 <0.5 disarmed ≥0.5 armed

## **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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency. The XTAL frequency is assumed to be 1 MHz, therefore the jumper on the CIO-CTR05 has to be in position 1 MHz not 5 MHz.

### **Relative output frequency**

Enter a value between 0 and  $.25 * \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 17.5 kHz, then choose F2=100kHz as the **Frequency base** and

# Measurement Computing CIO-CTR05 Counter PWM & ARM

---

enter 0.175 as the **Relative output frequency**.  $100 \text{ kHz} \times 0.175 = 17.5 \text{ kHz}$

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running. If the application is running, the second input port controls whether the counter is armed or disarmed. This parameter sets the TTL level when the counter is disarmed.

## **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

## **Base address**

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

0x300

# Measurement Computing CIO-CTR05 Counter FM

---

**Purpose** CIO-CTR05 Counter FM block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-CTR05 has one AM9513A chip with five counters.  
The CIO-CTR05 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 PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency. The XTAL frequency is assumed to be 1 MHz, therefore the jumper on the CIO-CTR05 has to 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.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---



## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and gets 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

# Measurement Computing CIO-CTR05 Counter FM & ARM

**Purpose** CIO-CTR05 Counter FM & ARM block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-CTR05 has one AM9513A chip with five counters.

The CIO-CTR05 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 to arm and disarm the counter by the second block input. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Variable frequency: Double Arm: Double	<0.5 disarmed ≥0.5 armed

## **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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency. The XTAL frequency is assumed to be 1 MHz, therefore the jumper on the CIO-CTR05 has to 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.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

### **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the **Output duty cycle** defined in the setting above define 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

# Measurement Computing CIO-CTR05 PWM Capture

---

**Purpose** CIO-CTR05 PWM Capture block

**Library** xPC Target Library for Measurement Computing

**Note** This block programs the AMD9513A for capturing PWM signals by using two counters. One counter measures the cycle duration, and the other counter measures the duration 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 has to enter the pins named GATE of both corresponding counter channels (parallel wiring). Both CLK pins have to be left unconnected.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	0 to 1

## Block Parameters

### Counter

From the list, choose 1&2, 2&3, 3&4, 4&5. This selects which two counters the driver block uses to determine the PWM. In each case, one block is needed for each counter.

### Frequency base

From the list, choose F1=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency. The XTAL frequency is assumed to be 1 MHz, therefore the jumper on the CIO-CTR05 has to 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)

# Measurement Computing CIO-CTR05 PWM Capture

---

## **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

# Measurement Computing CIO-CTR05 Frequency Capture

**Purpose** CIO-CTR05 Frequency Capture block

**Library** xPC Target Library for Measurement Computing

**Note** This block programs the AMD9513A 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 has to enter the pin named GATE of the corresponding counter channel. The CLK pin has to be left unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, 4, or 5. 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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency. The XTAL frequency is assumed to be 1 MHz, therefore the jumper on the CIO-CTR05 has to 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)

# Measurement Computing CIO-CTR05 Frequency Capture

---

## **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

# Measurement Computing CIO-CTRxx

---

<b>Purpose</b>	CIO-CTRxx block
<b>Library</b>	xPC Target Library for Measurement Computing
<b>Description</b>	You can use this block to program the AMD9513A counter. The PWM, PWM & ARM, FM, FM & ARM, PWM Capture, and Frequency Capture blocks use this block in their underlying subsystems. The API for this block is not currently documented.



## Board

Measurement Computing CIO-CTR10

## General Description

The CIO-CTR10 is an I/O board with ten counter/timer channels (16-bit).

It contains one AM9513A counter/timer chip. For additional information about the various counter/timer modes of that chip see the AM9513A data sheet which is part of the board documentation.

xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-CTR10 Counter PWM
- Measurement Computing CIO-CTR10 Counter PWM & ARM
- Measurement Computing CIO-CTR10 Counter FM
- Measurement Computing CIO-CTR10 Counter FM & ARM
- Measurement Computing CIO-CTR10 PWM Capture
- Measurement Computing CIO-CTR10 Frequency Capture
- Measurement Computing CIO-CTRxx

## Board Characteristics

Board name	CIO-CTR05
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing CIO-CTR10 Counter PWM

---

**Purpose** CIO-CTR10 Counter PWM

**Library** xPC Target Library for Measurement Computing

**Note** The CIOCTR10 has one AM9513A chip with ten counters.

The CIO-CTR10 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 which defines the variable duty cycle between 0 and 1. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## 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 CIO-CTR05 has to be in position 1 MHz not 5 MHz.

### Relative output frequency

Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 17.5 kHz, then choose F2=100kHz as the **Frequency base** and enter 0.175 as the **Relative output frequency**.  $100 \text{ kHz} \times 0.175 = 17.5 \text{ kHz}$

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and gets 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

# Measurement Computing CIO-CTR10 Counter PWM & ARM

---

**Purpose** CIO-CTR10 Counter PWM & ARM block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-CTR10 has two AM9513A chip with ten counters.

The CIO-CTR10 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 to arm and disarm the counter by 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

## 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 CIO-CTR05 has to be in position 1 MHz not 5 MHz.

### Relative output frequency

Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 17.5 kHz, then choose F2=100kHz as the **Frequency base** and

enter 0.175 as the **Relative output frequency**.  $100 \text{ kHz} \times 0.175 = 17.5 \text{ kHz}$

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running. If the application is running, the second input port controls whether the counter is armed or disarmed. This parameter sets the TTL level when the counter is disarmed.

## **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

## **Base address**

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

0x300

# Measurement Computing CIO-CTR10 Counter FM

---

**Purpose** CIO-CTR10 Counter FM block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-CTR10 has two AM9513A chip with ten counters.  
The CIO-CTR05 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 PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **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 CIO-CTR05 has to 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.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and gets 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

# Measurement Computing CIO-CTR10 Counter FM & ARM

**Purpose** CIO-CTR10 Counter FM & ARM block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-CTR10 has two AM9513A chips with ten counters.

The CIO-CTR10 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 to arm and disarm the counter by the second block input. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Variable frequency: Double Arm: Double	<0.5 disarmed ≥0.5 armed

## **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 CIO-CTR05 has to 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.



---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

### **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the **Output duty cycle** defined in the setting above define 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

# Measurement Computing CIO-CTR10 PWM Capture

---

**Purpose** CIO-CTR10 PWM Capture block

**Library** xPC Target Library for Measurement Computing

**Note** This block programs the AMD9513A for capturing PWM signals by using two counters. One counter measures the cycle duration, and the other counter measures the duration 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 has to enter the pins named GATE of both corresponding counter channels (parallel wiring). Both CLK pins have to be left unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1&2, 2&3, 3&4, 4&5, 5&6, 6&7, 7&8, 8&9, 9&10. This selects which two counters the driver block uses to determine the PWM. 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 CIO-CTR10 has to 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)

# Measurement Computing CIO-CTR10 PWM Capture

---

## **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

# Measurement Computing CIO-CTR10 Frequency Capture

**Purpose** CIO-CTR10 Frequency Capture block

**Library** xPC Target Library for Measurement Computing

**Note** This block programs the AMD9513A 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 has to enter the pin named GATE of the corresponding counter channel. The CLK pin has to be left unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **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 CIO-CTR10 has to 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).

# Measurement Computing CIO-CTR10 Frequency Capture

---

## **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

# Measurement Computing CIO-CTRxx

---

**Purpose**

CIO-CTRxx block

**Library**

xPC Target Library for Measurement Computing

**Note**

You can use this block to program the AMD9513A counter. The PWM, PWM & ARM, FM, FM & ARM, PWM Capture, and Frequency Capture blocks use this block in their underlying subsystems. The API for this block is not currently documented.

# Measurement Computing CIO-DAC08 (/12)

---

**Board** CIO-DAC08 (/12)

**General Description** The CIO-DAC08 (/12) is an I/O board with eight analog output (D/A) channels (12-bit).

xPC Target supports this board with this driver block:

- Measurement Computing CIO-DAC08 (/12) Analog Output (D/A)

<b>Board Characteristics</b>	Board name	CIO-DAC08 (CIO-DAC08/12)
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DAC08 (/12) Analog Output (D/A)

**Purpose** CIO-DAC08 (/12) Analog Output (D/A) block

**Library** xPC Target Library for Measurement Computing

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 8. This board 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

Range code for each of the channels in the channel vector. The range vector must have the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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



# Measurement Computing CIO-DAC08 (/12) Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch 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.

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAC08/16

---

**Board** Measurement Computing CIO-DAC08/16

**General Description** The CIO-DAC08/16 is an I/O board with 8 analog output (D/A) channels (16-bit).

xPC Target supports this board with this driver block:

- Measurement Computing CIO-DAC08/16 Analog Output (D/A)

<b>Board Characteristics</b>	Board name	CIO-DAC08/16
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DAC08/16 Analog Output (D/A)

**Purpose** CIO-DAC08/16 Analog Output (D/A) block

**Library** xPC Target Library for Measurement Computing

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 8. This board 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

Range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Measurement Computing CIO-DAC08/16 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch 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.

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAC16 (/12)

---

**Board** Measurement Computing CIO-DAC16 (/12)

**General Description** The CIO-DAC016 is an I/O board with 16 analog output (D/A) channels (12-bit).

xPC Target supports this board with one driver block:

- Measurement Computing CIO-DAC16 Analog Output (D/A)

<b>Board Characteristics</b>	Board name	CIO-DAC16 (/12)
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DAC16 Analog Output (D/A)

---

**Purpose** CIO-DAC16 Analog Output (D/A) block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Channel vector**

Enter 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 vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +10	10

# Measurement Computing CIO-DAC16 Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch 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.

## Sample time

Base sample time of a multiple of the base sample time.

## Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAC16 Analog Output (D/A)

---

The jumpers by the range DIP switches on the board all have to be in the XFER position. The Wait-State jumper has to be in the off position.



**Board** Measurement Computing CIO-DAC16/16

**General Description** The CIO-DAC16/16 is an I/O board with 16 analog output (D/A) channels (16-bit).

xPC Target supports this board with this driver block:

- Measurement Computing CIO-DAC16/16 Analog Output (D/A)

<b>Board Characteristics</b>	Board name	CIO-DAC08/16
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DAC16/16 Analog Output (D/A)

---

**Purpose** CIO-DAC16/16 Analog Output (D/A) block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This board allows the selection of individual A/D 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**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +10	10

# Measurement Computing CIO-DAC16/16 Analog Output (D/A)

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch 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.

## Sample time

Base sample time of a multiple of the base sample time.

## Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAC16/16 Analog Output (D/A)

---

The jumpers by the range DIP switches on the board all have to be in the XFER position. The Wait-State jumper has to be in the off position.

# Measurement Computing CIO-DAS16/330

---

**Board** Measurement Computing CIO-DAS16/330

**General Description** The CIO-DAS16/330 is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 330 kHz, 4 digital input lines, and 4 digital output lines.

xPC Target supports this board with one driver block:

- Measurement Computing CIO-DAS16/330 Analog Input (A/D)

---

**Note** xPC Target does not support the digital I/O on this board.

---

<b>Board Characteristics</b>	Board name	CIO-DAS16/330
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# Measurement Computing CIO-DAS16/330 Analog Input (A/D)

---

**Purpose** CIO-DAS16/330 Analog Input (A/D) block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of different range for each channel.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended (16 channels)
- Differential (8 channels)

This choice must correspond to the MUX-switch setting on the board.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Measurement Computing CIO-DAS16/330 Analog Input (A/D)

---

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAS16/JR (/12)

---

## Board

Measurement Computing CIO-DAS16/JR (/12)

## General Description

The CIO-DAS16/JR is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 130 kHz, 4 digital input lines, 4 digital output lines, and 3 counter/timers (16-bit). An external signal conditioning board can be added to the CIO-DAS16/JR board.

xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DAS16/JR Analog Input (A/D)
- Measurement Computing CIO-DAS16/JR (/12) Analog Input (A/D) with EXP

---

**Note** xPC Target does not support the digital I/O or counters on this board.

---

## Board Characteristics

Board name	CIO-DAS16/JR
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes



# Measurement Computing CIO-DAS16/JR Analog Input (A/D)

**Purpose** CIO-DAS16/JR Analog Input (A/D) block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of a different range for each channel.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended (16 channels)
- Differential (8 channels)

This choice must correspond to the MUX-switch setting on the board.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Measurement Computing CIO-DAS16/JR Analog Input (A/D)

---

## Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAS16/JR (/12) Analog Input (A/D) with EXP

**Purpose** CIO-DAS16/JR (/12) Analog Input with EXP Signal Conditioning Board block

**Library** xPC Target Library for Measurement Computing

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

There are signal conditioning boards (external devices) available from Measurement Computing which can be connected to the CIO-DAS16/JR. Each EXP-board contains its own multiplexer circuit which multiplexes a maximum number of 16 EXP-channels to one A/D-channel of the CIO-DAS16/JR. For this type of operation the CIO-DAS16/JR has to be setup for single-ended input mode and this results in a theoretical number of 256 EXP-channels per CIO-DAS/16JR board.

- EXP16
- EXP32
- EXP-BRIDGE16
- EXP-RTD
- EXP-GP

## **Block Parameters**

### **EXP Channel vector**

This parameter describes the EXP-channels used. Because always a group of 16 EXP-channels are mapped to one A/D-channel of the CIO-DAS16/JR the EXP-channel vector can contain elements between 0 and 15 and no value should occur twice. The number of elements of the vector defines the number of block outputs. The EXP-channel defined as the first element is output at the first block output, the EXP-channel defined as the second element is output at the second block output and so on.

# Measurement Computing CIO-DAS16/JR (/12) Analog Input (A/D) with EXP

---

Example: EXP Channel Vector: [4,0,12]  
the Signal of EXP-channel 4 is output at block outut 1  
the Signal of EXP-channel 0 is output at block outut 2  
the Signal of EXP-channel 12 is output at block outut 3

---

**Note** If a EXP32 is used and the EXP-channels 16 to 31 should be acquired, the elements of the EXP Channel Vector have still to be in the range of 0 to 15. Therefore the EXP-channel numbers have to be subsaturated by the constant 16.

---

A special case is provided by setting the EXP Channel Vector to an empty vector. In this case it is assumed that no EXP-board is connected to the specified A/D-channel (see dialog field A/D Board Channel) and the signal is directly connected to the A/D-input of the CIO-DAS16Jr board. This feature allows to use the A/D-channels of a CIO-DAS16Jr either for EXP-channels or for direct input. Therefore, it is not necessary to purchase another A/D-board for direct input.

---

**Note** This feature should only be used if at least one EXP-board has to be connected to the CIO-DAS16Jr. If all inputs are directly connected to the A/D board use the CIO-DAS16Jr/12 (2.2.1) driver instead which allows much higher sample rates.

---

## EXP Gain

This parameter describes the gains for each EXP-channel used. This vector corresponds over his indices with the EXP-gain vector and must therefore have the same length. Because this I/O-driver can be used together with all different EXP-boards there is no restriction about the gain value itself. The EXP-board manual should be contacted to know what the gains of the different EXP-boards are. The gains on the EXP-board depend on several DIP switches on the specific EXP-board.

# Measurement Computing CIO-DAS16/JR (/12) Analog Input (A/D) with EXP

---

Example:

EXP Channel Vector: [4,0,12]

EXP Gain Vector: [1,1000,200]

EXP-channel 4 has gain 1, channel 0 gain 1000 and channel 12 gain 200

If EXP Channel Vector is an empty vector EXP Gain Vector has to be an empty vector as well.

## A/D Board Channel

This field specifies to which A/D-channel of the CIO-DAS16Jr the block of 16 EXP-channels are mapped. Because the input coupling of the A/D board has to be single-ended channel 0 to 16 can be used. The channel selection jumpers on the EXP-boards have to be set accordingly to this software setting.

## A/D Board Range

This field specifies the input voltage range for the CIO-DAS16/JR which is the same for all 16 single-ended channels.

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of different range for each channel.

## Sample time

Enter the base sample time or a multiple of the base sample time.

## Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

---

**Note** If this driver is used the input coupling switch on the CIO-DAS16Jr has always to be in the 16 (single-ended) position.

---

# Measurement Computing CIO-DAS16JR/16

---

## Board

Measurement Computing CIO-DAS16JR/16

## General Description

The CIO-DAS16JR/16 is an I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 100 kHz, 4 digital input lines, 4 digital output lines and 3 counter/timers.

xPC Target supports this board with this driver block:

- Measurement Computing CIO-DAS16JR/16 Analog Input (A/D)

---

**Note** xPC Target does not support the digital I/O or the counters on this board.

---

## Board Characteristics

Board name	CIO-DAS16JR/16
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Measurement Computing CIO-DAS16JR/16 Analog Input (A/D)

---

**Purpose** CIO-DAS16JR/16 Analog Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of a different range for each channel.

### **Input coupling**

From the list, select one of the input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

This choice must correspond to the MUX-switch setting on the board.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Measurement Computing CIO-DAS16JR/16 Analog Input (A/D)

---

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



# Measurement Computing CIO-DAS1601/12

---

## Board

Measurement Computing CIO-DAS1601/12

## General Description

The CIO-DAS1601/12 is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sampling rate of 160 kHz, 2 analog output (D/A) channels (12-bit), 32 digital input and output lines, and 3 counters (16-bit).

xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DAS1601/12 Analog Input (A/D)
- Measurement Computing CIO-DAS1601/12 Analog Output (D/A)
- Measurement Computing CIO-DAS1601/12 Digital Input
- Measurement Computing CIO-DAS1601/12 Digital Output

---

**Note** xPC Target supports only 24 digital I/O lines and does not support the counters on this board.

---

## Board Characteristics

Board name	CIO-DAS1601/12
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	A/D: No, D/A: Yes, Digital I/O: Yes
Multiple board support	Yes

# Measurement Computing CIO-DAS1601/12 Analog Input (A/D)

---

**Purpose** CIO-DAS1601/12 Analog Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of a different range for each channel.

If a bipolar range is used, the bipolar switch on the board must be in the bipolar position. if a unipolar range is used, the bipolar switch must be in the unipolar position.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

# Measurement Computing CIO-DAS1601/12 Analog Input (A/D)

---

This choice must correspond to the MUX-switch setting on the board.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAS1601/12 Analog Output (D/A)

---

**Purpose** CIO-DAS1601/12 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## Block Parameters

### Channel vector

Enter 1 or 2. 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

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Measurement Computing CIO-DAS1601/12 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be placed according to the ranges used.

## **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**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAS1601/12 Digital Input

**Purpose** CIO-DAS1601/12 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/12 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DAS1601/12 Digital Input

---

**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

# Measurement Computing CIO-DAS1601/12 Digital Output

---

**Purpose** CIO-DAS1601/12 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/12 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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

# Measurement Computing CIO-DAS1602/12

---

## Board

Measurement Computing CIO-DAS1602/12

## General Description

The CIO-DAS1602/12 is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sampling rate of 100 kHz, 2 analog output (D/A) channels (12-bit), 32 digital input and output lines, and 3 counters (16-bit).

xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DAS1602/12 Analog Input (A/D)
- Measurement Computing CIO-DAS1602/12 Analog Output (D/A)
- Measurement Computing CIO-DAS1602/12 Digital Input
- Measurement Computing CIO-DAS1602/12 Digital Output

---

**Note** xPC Target supports only 24 digital I/O lines and does not support the counters on this board.

---

## Board Characteristics

Board Name	CIO-DAS1602/12
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	A/D: No, D/A: Yes, Digital I/O: Yes
Multiple board support	Yes

# Measurement Computing CIO-DAS1602/12 Analog Input (A/D)

---

**Purpose** CIO-DAS1602/12 Analog Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of a different range for each channel.

If a bipolar range is used, the bipolar switch on the board must be in the bipolar position. If a unipolar range is used, the bipolar switch must be in the unipolar position.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

This choice must correspond to the MUX-switch setting on the board.

# Measurement Computing CIO-DAS1602/12 Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAS1602/12 Analog Output (D/A)

**Purpose** CIO-DAS1602/12 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Measurement Computing CIO-DAS1602/12 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be planted according to the ranges used.

## **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**

Base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAS1602/12 Digital Input

**Purpose** CIO-DAS1602/12 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/12 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DAS1602/12 Digital Input

---

## **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



# Measurement Computing CIO-DAS1602/12 Digital Output

**Purpose** CIO-DAS1602/12 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/12 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DAS1602/12 Digital Output

---

## **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

## Board

Measurement Computing CIO-DAS1602/16

The CIO-DAS1602/16 is an I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit) with a maximum sampling rate of 100 kHz, 2 analog output (D/A) channels (16-bit), 32 digital I/O lines, and 3 counters (16-bit).

xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DAS1602/16 Analog Input (A/D)
- Measurement Computing CIO-DAS1602/16 Analog Output (D/A)
- Measurement Computing CIO-DAS 1602/16 Digital Input
- Measurement Computing CIO DAS1602/16 Digital Output

---

**Note** xPC Target supports only 24 digital I/O lines and does not support the counters on this board.

---

## Board Characteristics

Board Name	CIO-DAS1602/16
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	A/D: No, D/A: Yes, Digital I/O: Yes
Multiple board support	Yes

# Measurement Computing CIO-DAS1602/16 Analog Input (A/D)

---

**Purpose** CIO-DAS1602/16 Analog Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of a different range for each channel.

If a bipolar range is used, the bipolar switch on the board must be in the bipolar position. if a unipolar range is used, the bipolar switch must be in the unipolar position.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Measurement Computing CIO-DAS1602/16 Analog Input (A/D)

---

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAS1602/16 Analog Output (D/A)

**Purpose** CIO-DAS1602/16 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Measurement Computing CIO-DAS1602/16 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be planted according to the ranges used.

## **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**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DAS 1602/16 Digital Input

**Purpose** CIO-DAS 1602/16 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/16 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

### Hardware Input

TTL

### Block Output Data Type

Double

### Scaling

TTL low = 0.0

TTL high = 1.0

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.



# Measurement Computing CIO-DAS 1602/16 Digital Input

---

## **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

# Measurement Computing CIO DAS1602/16 Digital Output

---

**Purpose** CIO DAS1602/16 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/16 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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

# Measurement Computing CIO-DDA06 (/12)

---

**Board** Measurement Computing CIO-DDA06 (/12)

**General Description** The CIO-DDA06 (/12) is an I/O board with 6 analog output (D/A) channels (12-bit), and 24 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DDA06 (/12) Analog Output (D/A)
- Measurement Computing CIO-DDA06 (/12) Digital Input
- Measurement Computing CIO-DDA06 (/12) Digital Output

<b>Board Characteristics</b>	Board name	CIO-DDA06 (/12)
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DDA06 (/12) Analog Output (D/A)

**Purpose** CIO-DDA06 (/12) Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 6. 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**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.67 to +1.67	-1.67
-0.625 to +0.625	-0.625

# Measurement Computing CIO-DDA06 (/12) Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +10	10
0 to +5	5
0 to +2.5	2.5
0 to +1.67	1.67

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. The jumpers by the range DIP switches on the board all must be in the XFER position. The wait state jumper must be in the off position.

## **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**

Base sample time of a multiple of the base sample time.

# Measurement Computing CIO-DDA06 (/12) Analog Output (D/A)

---

## Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DDA06 (/12) Digital Input

**Purpose** CIO-DDA06 (/12) Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DDA06 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.



# Measurement Computing CIO-DDA06 (/12) Digital Input

---

## **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

# Measurement Computing CIO-DDA06 (/12) Digital Output

---

**Purpose** CIO-DDA06 (/12) Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DDA06 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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

# Measurement Computing CIO-DDA06/16

---

**Board** Measurement Computing CIO-DDA06/16

**General Description** The CIO-DDA06/16) is an I/O board with 6 analog output (D/A) channels (16-bit), and 24 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DDA06/16 Analog Output (D/A)
- Measurement Computing CIO-DDA06/16 Digital Input
- Measurement Computing CIO-DDA06/16 Digital Output

<b>Board Characteristics</b>	Board name	CIO-DDA06/16
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DDA06/16 Analog Output (D/A)

**Purpose** CIO-DDA06/16 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 6. 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

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.67 to +1.67	-1.67
-0.625 to +0.625	-0.625

# Measurement Computing CIO-DDA06/16 Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +10	10
0 to +5	5
0 to +2.5	2.5
0 to +1.67	1.67

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. The jumpers by the range DIP switches on the board all must be in the XFER position. The wait state jumper has to be in the off position.

## **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**

Base sample time of a multiple of the base sample time.

# Measurement Computing CIO-DDA06/16 Analog Output (D/A)

---

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-DDA06/16 Digital Input

**Purpose** CIO-DDA06/16 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DDA06/16 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.



# Measurement Computing CIO-DDA06/16 Digital Input

---

## **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

# Measurement Computing CIO-DDA06/16 Digital Output

**Purpose** CIO-DDA06/16 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DDA06/16 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

$< 0.5 = \text{TTL low}$

$\geq 0.5 = \text{TTL high}$

## 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 of 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 a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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

# Measurement Computing CIO-DIO24

---

**Board** Measurement Computing CIO-DIO24

**General Description** The CIO-DIO24 is an I/O board with 24 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DIO24 Digital Input
- Measurement Computing CIO-DIO24 Digital Output
- Measurement Computing CIO-DIO24 Signal Conditioning

<b>Board Characteristics</b>	Board name	CIO-DIO24
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DIO24 Digital Input

**Purpose** CIO-DIO24 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO24 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

### Hardware Input

TTL

### Block Output Data Type

Double

### Scaling

TTL low = 0.0

TTL high = 1.0

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DIO24 Digital Input

---

## **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

# Measurement Computing CIO-DIO24 Digital Output

**Purpose** CIO-DIO24 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO24 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DIO24 Digital Output

---

## **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



# Measurement Computing CIO-DIO24 Signal Conditioning

---

**Purpose**

CIO-DIO24 Signal Conditioning block

**Library**

xPC Target Library for Measurement Computing

**Block Parameters****Genix initialization file (path\file)**

Provide the filename of the Genix initialization file.

**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

# Measurement Computing CIO-DIO24H

---

**Board** Measurement Computing CIO-DIO24H

**General Description** The CIO-DIO24H is an I/O board with 24 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DIO24H Digital Input
- Measurement Computing CIO-DIO24H Digital Output

<b>Board Characteristics</b>	Board name	CIO-DIO24H
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DIO24H Digital Input

**Purpose** CIO-DIO24H Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO24H has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 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.

# Measurement Computing CIO-DIO24H Digital Input

---

## **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

# Measurement Computing CIO-DIO24H Digital Output

**Purpose** CIO-DIO24H Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO24H has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

$< 0.5 =$  TTL low  
 $\geq 0.5 =$  TTL high

## 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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DIO24H Digital Output

---

## **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

**Board** Measurement Computing CIO-DIO48

**General Description** The CIO-DIO48 is an I/O board with 48 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DIO48 Digital Input
- Measurement Computing CIO-DIO48 Digital Output

<b>Board Characteristics</b>	Board name	CIO-DIO48
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DIO48 Digital Input

---

**Purpose** CIO-DIO48 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO48 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.



# Measurement Computing CIO-DIO48 Digital Input

---

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **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

# Measurement Computing CIO-DIO48 Digital Output

---

**Purpose** CIO-DIO48 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO48 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **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

# Measurement Computing CIO-DIO48H

---

**Board** Measurement Computing CIO-DIO48H

**General Description** The CIO-DIO48H is an I/O board with 48 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DIO48H Digital Input
- Measurement Computing CIO-DIO48H Digital Output

<b>Board Characteristics</b>	Board name	CIO-DIO48H
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DIO48H Digital Input

**Purpose** CIO-DIO48H Digital Input block

**Library** xPC Target Library for Measurement Computing

**General Description** The CIO-DIO48H has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DIO48H Digital Input

---

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **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

# Measurement Computing CIO-DIO48H Digital Output

**Purpose** CIO-DIO48H Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO48H has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DIO48H Digital Output

---

## **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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **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



**Board** Measurement Computing CIO-DIO96

**General Description** The CIO-DIO96 is an I/O board with 96 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DIO96 Digital Input
- Measurement Computing CIO-DIO96 Digital Output

<b>Board Characteristics</b>	Board name	CIO-DIO96
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DIO96 Digital Input

---

**Purpose** CIO-DIO96 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO96 has four 8255 chips (1,2,3,4). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DIO96 Digital Input

---

## **Chip**

From the list choose 1, 2, 3, or 4. The **Chip** parameter defines which of the four 8255 chips is used.

## **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

# Measurement Computing CIO-DIO96 Digital Output

**Purpose** CIO-DIO96 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO96 has four 8255 chips (1,2,3,4). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

**Chip**

From the list choose 1, 2, 3, or 4. The **Chip** parameter defines which of the four 8255 chips is used.

**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

# Measurement Computing CIO-DIO192

---

**Board** Measurement Computing CIO-DIO192

**General Description** The CIO - DIO192 is an I/O board with 192 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-DIO192 Digital Input
- Measurement Computing CIO-DIO192 Digital Output

<b>Board Characteristics</b>	Board name	CIO-DIO192
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DIO192 Digital Input

**Purpose** CIO-DIO192 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO96 has eight 8255 chips (1,2,3,4,5,6,7,8). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DIO192 Digital Input

---

## **Chip**

From the list choose 1, 2, 3, 4, 5, 6, 7, or 8. The **Chip** parameter defines which of the eight 8255 chips is used.

## **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



# Measurement Computing CIO-DIO192 Digital Output

**Purpose** CIO-DIO192 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO192 has eight 8255 chips (1,2,3,4,5,6,7,8). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

$< 0.5 = \text{TTL low}$

$\geq 0.5 = \text{TTL high}$

## 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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing CIO-DIO192 Digital Output

---

## **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.

## **Chip**

From the list choose 1, 2, 3, 4, 5, 6, 7, or 8. The **Chip** parameter defines which of the eight 8255 chips is used.

## **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

**Board** Measurement Computing CIO-DO24DD

**General Description** The CIO-DO24DD is an I/O board with 24 open-collector digital output lines.

xPC Target supports this board with this driver block:

- Measurement Computing CIO-DO24DD Digital Output

<b>Board Characteristics</b>	Board name	CIO-DO24DD
	Manufacturer	Measurement Computing
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing CIO-DO24DD Digital Output

**Purpose** CIO-DO24DD Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO24DD has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that are configured as outputs.  
Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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

# Measurement Computing CIO-PDISO16

---

## Board

Measurement Computing CIO-PDISO16

## General Description

The CIO-PDISO16 is an I/O board with 16 isolated digital input lines and 16 relay driven digital output lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing CIO-PDISO16 Digital Input
- Measurement Computing CIO-PDISO16 Digital Output

---

**Note** xPC Target does not support the 16 relays on this board.

---

## Board Characteristics

Board name	CIO-PDISO16
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing CIO-PDISO16 Digital Input

**Purpose** CIO-PDISO16 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-PDISO16 has two independent connectors. Each connector has 8 digital input lines.

Use a separate driver block for each connector.

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
5 to 24 volts DC/AC	Double	~0 volts = 0.0 5 to 24 volts = 1.0

## Block Parameters

### Number of Channels

Enter a number between 1 and 8 to select the number of digital input lines used with this connector. This driver does not allow the selection of individual digital input lines.

### Section(Connector)

From the list, choose either 1 (nearest to backplate) or 2 (farthest from backplate to select the connector used).

### Sample time

Enter the base sample time or a multiple of the base sample time.

### Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

The Wait-State jumper has to be in the off position.

The switch and jumper settings, that are not mentioned here, have no influence on the running of xPC Target.

# Measurement Computing CIO-PDISO16 Digital Output

---

**Purpose** CIO-PDISO16 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-PDISO16 has two independent connectors. Each connector has 8 relay driven digital input lines.

Use a separate driver block for each connector.

## Scaling Input to Output

### Hardware Output

relay

### Block Input Data Type

Double

### Scaling

< 0.5 =  
Relay open  
≥ 0.5 = Relay  
closed

## Block Parameters

### Number of Channels

Enter a number between 1 and 8 to select the number of digital output lines used with this connector. This driver does not allow the selection of individual digital output lines.

### Section(Connector)

From the list, choose either 1 (nearest to backplate) or 2 (farthest from backplate to select the connector used).

### Sample time

Enter the base sample time or a multiple of the base sample time.

### Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

The Wait-State jumper has to be in the off position.



# Measurement Computing CIO-PDISO16 Digital Output

---

The switch and jumper settings, that are not mentioned here, have no influence on the running of xPC Target.

# Measurement Computing CIO-QUAD02

---

## Board

Measurement Computing CIO-QUAD02

## General Description

The CIO-QUAD02 is a 24-bit counting board with 2 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.

xPC Target supports this board with this driver block:

- Measurement Computing CIO-QUAD02 Incremental Encoder

See Measurement Computing CIO-QUAD02 Incremental Encoder if you have an earlier version of this driver.

## Board Characteristics

Board name	CIO-QUAD02
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing CIO-QUAD02 Incremental Encoder

---

**Purpose** CIO-QUAD02 Incremental Encoder block

**Library** xPC Target Library for Measurement Computing

**Note** This driver block has one block output: **Count**.

The raw count value is a 24-bit integer that is converted to a double precision float output from the block. The 24-bit value can be interpreted in different ways (see **Count Range**).

In this section, the clockwise direction is the direction in which the counter increments. The counterclockwise direction is the direction in which the counter decrements. Note that the counter direction depends on the encoder manufacturer and the way in which the encoder is installed.

---

**Note** If you are using an earlier version of this driver block, see Measurement Computing CIO-QUAD02 Incremental Encoder.

---

## Block Parameters

### Channel

From the list, choose 1, 2, 3, or 4. This parameter specifies the channel you use for this block. For the same board (same base address) two blocks cannot have the same channel number.

### Index input resets counter

Choose this check box to control the behavior when the index is reached. For example, a rotary encoder typically reaches the index at one particular position in its range of motion. Setting this check box to reset the count value allows you to get a repeatable position from an encoder. When the encoder reaches the index (asserts the index), the counter value is set to the value given in the **Count Limit or Reset Value** field.

# Measurement Computing CIO-QUAD02 Incremental Encoder

---

## Index Polarity

From the list, select the index pulse slope that the board should detect. Select Positive Index for a positive slope; select Negative Index for a negative slope.

## Counting Mode

From the list, select a counting mode (Normal, Range Limit, Non-recycle, or Modulo-N). The output of the block depends on both the setting of **Index input resets counter** and on the counting mode as described in the following table.

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Normal	<p>Upon model start, the output count is initialized to <b>Count Limit or Reset Value</b>. The output count increments or decrements one full revolution from the <b>Count Limit or Reset Value</b>. The output count is reset to this value each time the index is reached.</p> <p>For example, if <b>Index input resets counter</b> is selected, and a rotary encoder has 1024 counts per revolution, and a <b>Count Limit or Reset Value</b> value of 5000, a full clockwise revolution of the encoder increments the counter to 6023, at which the index is reached. The counter then resets to 5000.</p>	<p>Upon model start, the counter is initialized to 0. The counter then increments or decrements until it either overflows or underflows. For a rotary encoder, the number of revolutions is the number of output counts divided by the number of counts per revolution. The resulting integer is the number of revolutions; the fractional part can be converted to an angle.</p>

# Measurement Computing CIO-QUAD02 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Range Limit	<p>Upon model start, the counter increments or decrements until it reaches the count limit <b>Count Limit or Reset Value</b>. If the clock motion is clockwise, the counter stays at <b>Count Limit or Reset Value</b> once it reaches that value. If the clock motion is counterclockwise, the counter decrements one revolution then resets to <b>Count Limit or Reset Value</b>.</p>	<p>Upon model start, the counter is initialized to 0. In the clockwise direction, the counter increments until it saturates at <b>Count Limit or Reset Value</b>. In the counterclockwise direction, the counter decrements until it saturates at 0.</p> <p>You can use this setting to set a repeatable 0 when there is a mechanical stop. At model start, the count is initialized to 0. The machine then shifts to the negative direction stop. At that point, the encoder reports 0 at the stop and all other positions will be positive and repeatable.</p>

# Measurement Computing CIO-QUAD02 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Non-recycle	Upon model start, the counter is initialized to <b>Count Limit or Reset Value</b> . With counterclockwise rotation, the counter decrements and saturates at 0. If the encoder reaches the index before the counter saturates at 0, the counter resets to <b>Count Limit or Reset Value</b> . With clockwise rotation, the counter increments until the encoder reaches the index. When the encoder reaches the index, the counter resets to <b>Count Limit or Reset Value</b> . You can use this setting to place 0 at the first index inside a mechanical limit.	Upon model start, the counter decrements until it reaches either 0 or <b>Count Limit or Reset Value</b> . The counter restarts when a reset event occurs, such as the index has caused a reset. The behavior of this option is undefined.

# Measurement Computing CIO-QUAD02 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Modulo-N	The behavior of this option is undefined.	Upon model start, the counter is initialized to 0. In the clockwise direction, the counter increments to <b>Count Limit or Reset Value</b> , then resets to 0 and repeats. When the counter decrements, it counts to 0, then resets to the <b>Count Limit or Reset Value</b> value and continues to decrement. You can use this setting to repeat intervals that do not match the distance between indexes (resulting, for example, from gear ratios).
Non-recycle	Upon model start, the counter is initialized to <b>Count Limit or Reset Value</b> . With counterclockwise rotation, the counter decrements and saturates at 0. If the encoder reaches the index before the counter saturates at 0, the counter resets to <b>Count Limit or Reset Value</b> . With clockwise rotation, the counter increments until the encoder reaches the index. When the encoder reaches the index, the counter resets to <b>Count Limit or Reset Value</b> . You can use this setting to place 0 at the first index inside a mechanical limit.	The behavior of this option is undefined.

# Measurement Computing CIO-QUAD02 Incremental Encoder

---

## Count Limit or Reset Value

The preset register (PR) on the counter chip is a 24-bit quantity. The counter uses this value as a limit for the **Range Limit**, **Non-recycle**, and **Modulo-N** counting modes.

When **Index input resets counter** is selected, this value acts as both the initial value at model start and the value that an index resets the count to.

You can enter negative limit values, but the counter treats them as large positive values. For example, a limit of -1 results in the **Modulo-N** range from 0 to  $2^{24} - 1$ , not from -1 to 0.

## Count Range

From the list, choose  $0 \dots 2^{24} - 1$  or  $-2^{23} \dots +2^{23} - 1$ . The counter uses a 24-bit accumulator which overflows from  $2^{24} - 1$  to 0. This same overflow can be interpreted as the transition from -1 to 0. Selecting the unipolar range 0 to  $2^{24} - 1$  puts the discontinuity in counts at 0. Selecting the bipolar range of  $(-2)^{23}$  to  $2^{23} - 1$  puts the discontinuity far away from 0. The bipolar range is achieved by sign extending the 24-bit count to 32 bits before converting to double.

## Count Speed

From the list, choose non-quadrature, 1X, 2X, or 4X:

- non-quadrature — Counts the rising edges of the A input, losing direction information in the process. The presence of a quadrature signal on the B input might cause undefined results. You should disconnect the B and index inputs.
- 1X — Counts up or down once per complete cycle of the quadrature signal.
- 2X — Counts up or down twice per complete cycle of the quadrature signal.
- 4X — Counts up or down four times per complete cycle of the quadrature signal.



# Measurement Computing CIO-QUAD02 Incremental Encoder

---

## **Filter prescale factor**

Enter a base prescale factor for digital filtering. This filter helps eliminate high frequency noise. Enter a value from 0 to ff in hexadecimal (0 to 256 in decimal). For example, for a prescale factor value of ff in hexadecimal, enter

0xff

## **Sample Time**

Enter a 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). Enter -1 to inherit the model sample time from another block. The sample time is in seconds.

## **Base Address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-QUAD04

---

## Board

Measurement Computing CIO-QUAD04

## General Description

The CIO-QUAD04 is a 24-bit counting board with 4 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.

xPC Target supports this board with this driver block:

- Measurement Computing CIO-QUAD04 Incremental Encoder

See Measurement Computing CIO-QUAD04 Incremental Encoder if you have an earlier version of this driver.

Note on usage:

There is a hardware limitation with the CIO-Quad04 boards in the way that they handle unconnected inputs. Even with the **Index input resets counter** check box disabled, noise on the index input line might cause the counter chip on the board to latch an incorrect value. This value differs from the correct value by a multiple of 256. The frequency of occurrence of the bad reads depends on the count frequency coming from the encoder. High count rates result in more bad reads.

To work around this issue, try the following:

- 1** If the index signal from an encoder is not being used, connect the index inputs.
- 2** Connect the index+ input to +5 volts.
- 3** Connect the index- input to ground.

These connections should be made as close to the board as possible to minimize any crosstalk at high count frequencies.

## **Board Characteristics**

Board name	CIO-QUAD04
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing CIO-QUAD04 Incremental Encoder

---

**Purpose** CIO-QUAD04 Incremental Encoder block

**Library** xPC Target Library for Measurement Computing

**Note** This driver block has one block output: **Count**.

The raw count value is a 24-bit integer that is converted to a double precision float output from the block. The 24-bit value can be interpreted in different ways (see **Count Range**).

In this section, the clockwise direction is the direction in which the counter increments. The counterclockwise direction is the direction in which the counter decrements. Note that the counter direction depends on the encoder manufacturer and the way in which the encoder is installed.

---

**Note** If you are using an earlier version of this driver block, see Measurement Computing CIO-QUAD04 Incremental Encoder.

---

## Block Parameters

### Channel

From the list, choose 1, 2, 3, or 4. This parameter specifies the channel you use for this block. For the same board (same base address) two blocks cannot have the same channel number.

### Index input resets counter

Choose this check box to control the behavior when the index is reached. For example, a rotary encoder typically reaches the index at one particular position in its range of motion. Setting this check box to reset the count value allows you to get a repeatable position from an encoder. When the encoder reaches the index (asserts the index), the counter value is set to the value given in the **Count Limit or Reset Value** field.

# Measurement Computing CIO-QUAD04 Incremental Encoder

## Index Polarity

From the list, select the index pulse slope that the board should detect. Select **Positive Index** for a positive slope; select **Negative Index** for a negative slope.

## Counting Mode

From the list, select a counting mode (Normal, Range Limit, Non-recycle, or Modulo-N). The output of the block depends on both the setting of **Index input resets counter** and on the counting mode as described in the following table.

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Normal	<p>Upon model start, the output count is initialized to <b>Count Limit or Reset Value</b>. The output count increments or decrements one full revolution from the <b>Count Limit or Reset Value</b>. The output count is reset to this value each time the index is reached.</p> <p>For example, if <b>Index input resets counter</b> is selected, and a rotary encoder has 1024 counts per revolution, and a <b>Count Limit or Reset Value</b> value of 5000, a full clockwise revolution of the encoder increments the counter to 6023, at which the index is reached. The counter then resets to 5000.</p>	<p>Upon model start, the counter is initialized to 0. The counter then increments or decrements until it either overflows or underflows. For a rotary encoder, the number of revolutions is the number of output counts divided by the number of counts per revolution. The resulting integer is the number of revolutions; the fractional part can be converted to an angle.</p>

# Measurement Computing CIO-QUAD04 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Range Limit	<p>Upon model start, the counter increments or decrements until it reaches the count limit <b>Count Limit or Reset Value</b>. If the clock motion is clockwise, the counter stays at <b>Count Limit or Reset Value</b> once it reaches that value. If the clock motion is counterclockwise, the counter decrements one revolution then resets to <b>Count Limit or Reset Value</b>.</p>	<p>Upon model start, the counter is initialized to 0. In the clockwise direction, the counter increments until it saturates at <b>Count Limit or Reset Value</b>. In the counterclockwise direction, the counter decrements until it saturates at 0.</p> <p>You can use this setting to set a repeatable 0 when there is a mechanical stop. At model start, the count is initialized to 0. The machine then shifts to the negative direction stop. At that point, the encoder reports 0 at the stop and all other positions will be positive and repeatable.</p>

# Measurement Computing CIO-QUAD04 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Non-recycle	<p>Upon model start, the counter is initialized to <b>Count Limit or Reset Value</b>. With counterclockwise rotation, the counter decrements and saturates at 0. If the encoder reaches the index before the counter saturates at 0, the counter resets to <b>Count Limit or Reset Value</b>. With clockwise rotation, the counter increments until the encoder reaches the index. When the encoder reaches the index, the counter resets to <b>Count Limit or Reset Value</b>. You can use this setting to place 0 at the first index inside a mechanical limit.</p>	<p>The behavior of this option is undefined.</p>
Modulo-N	<p>The behavior of this option is undefined.</p>	<p>Upon model start, the counter is initialized to 0. In the clockwise direction, the counter increments to <b>Count Limit or Reset Value</b>, then resets to 0 and repeats. When the counter decrements, it counts to 0, then resets to the <b>Count Limit or Reset Value</b> value and continues to decrement. You can use this setting to repeat intervals that do not match the distance between indexes (resulting, for example, from gear ratios).</p>

# Measurement Computing CIO-QUAD04 Incremental Encoder

---

## Count Limit or Reset Value

The preset register (PR) on the counter chip is a 24-bit quantity. The counter uses this value as a limit for the **Range Limit**, **Non-recycle**, and **Modulo-N** counting modes.

When **Index input resets counter** is selected, this value acts as both the initial value at model start and the value that an index resets the count to.

You can enter negative limit values, but the counter treats them as large positive values. For example, a limit of -1 results in the **Modulo-N** range from 0 to  $2^{24} - 1$ , not from -1 to 0.

## Count Range

From the list, choose  $0 \dots 2^{24} - 1$  or  $-2^{23} \dots +2^{23} - 1$ . The counter uses a 24-bit accumulator which overflows from  $2^{24} - 1$  to 0. This same overflow can be interpreted as the transition from -1 to 0. Selecting the unipolar range 0 to  $2^{24} - 1$  puts the discontinuity in counts at 0. Selecting the bipolar range of  $(-2)^{23}$  to  $2^{23} - 1$  puts the discontinuity far away from 0. The bipolar range is achieved by sign extending the 24-bit count to 32 bits before converting to double.

## Count Speed

From the list, choose non-quadrature, 1X, 2X, or 4X:

- non-quadrature — Counts the rising edges of the A input, losing direction information in the process. The presence of a quadrature signal on the B input might cause undefined results. You should disconnect the B and index inputs.
- 1X — Counts up or down once per complete cycle of the quadrature signal.
- 2X — Counts up or down twice per complete cycle of the quadrature signal.
- 4X — Counts up or down four times per complete cycle of the quadrature signal.



# Measurement Computing CIO-QUAD04 Incremental Encoder

---

## **Filter prescale factor**

Enter a base prescale factor for digital filtering. This filter helps eliminate high frequency noise. Enter a value from 0 to ff in hexadecimal (0 to 256 in decimal). For example, for a prescale factor value of ff in hexadecimal, enter

0xff

## **Sample time**

Enter a 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). Enter -1 to inherit the model sample time from another block. The sample time is in seconds.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing PC104-DAC06 (/12)

---

**Board** Measurement Computing PC104-DAC06 (/12)

**General Description** The PC104-DAC06 (12) is an I/O board with 6 analog output (D/A) channels (12-bit).

xPC Target supports this board with this driver block:

- Measurement Computing PC104-DAC06 (/12) Analog Output (D/A)

<b>Board Characteristics</b>	Board name	PC104-DAC06 (/12)
	Manufacturer	Measurement Computing
	Bus type	PC/104
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# Measurement Computing PC104-DAC06 (/12) Analog Output (D/A)

**Purpose** PC104-DAC06 (/12) Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 6. This board 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 channels, enter

[1,2]

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

### Range vector

Range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Measurement Computing PC104-DAC06 (/12) Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to 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.

## **Sample time**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

The jumpers by the range DIP switches on the board all have to be in the XFER position. The Wait-State jumper has to be in the off position.

# Measurement Computing PC104-DAS16JR/12

---

**Board** Measurement Computing PC104-DAS16JR/12

## General Description

The PC104-DAS16JR/12 is an I/O board with 16 single or 8 differential analog input channels (12-bit) with a maximum sample rate of 150 kHz, 4 digital input lines and 4 digital output lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PC104-DAS16JR/12 Analog Input (A/D)
- Measurement Computing PC104-DAS16JR/12 Digital Input
- Measurement Computing PC104-DAS16JR/12 Digital Output

## Board Characteristics

Board name	PC104-DAS16JR/12
Manufacturer	Measurement Computing
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Measurement Computing PC104-DAS16JR/12 Analog Input (A/D)

---

**Purpose** PC104-DAS16JR/12 Analog Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either  $\pm 10V$  (-10 volts to +10 volts),  $\pm 5V$ ,  $\pm 2.5V$ ,  $\pm 1.25V$ ,  $\pm 0.625V$ , 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of a different range for each channel.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

This choice must correspond to the MUX-switch setting on the board.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Measurement Computing PC104-DAS16JR/12 Analog Input (A/D)

---

## Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing PC104-DAS16JR/12 Digital Input

---

**Purpose** PC104-DAS16JR/12 Digital Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

Enter a number between 1 and 4 to select the number of digital input lines used. This driver does not allow the selection of individual digital input lines.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



# Measurement Computing PC104-DAS16JR/12 Digital Output

**Purpose** PC104-DAS16JR/12 Digital Output block

**Library** xPC Target Library for Measurement Computing

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Number of Channels**

Enter a number between 1 and 4 to select the number of digital output lines used. This driver does not allow the selection of individual digital output lines.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

The switch and jumper settings, that are not mentioned here, have no influence on the running of xPC Target.

# Measurement Computing PC104-DAS16JR/16

---

## Board

Measurement Computing PC104-DAS16JR/16

## General Description

The PC104-DAS16JR/16 is an I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 100 kHz, 4 digital input lines and 4 digital output lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PC104-DAS16JR/16 Analog Input (A/D)
- Measurement Computing PC104-DAS16JR/16 Digital Input
- Measurement Computing PC104-DAS16JR/16 Digital Output

## Board Characteristics

Board name	PC104-DAS16JR/16
Manufacturer	Measurement Computing
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Measurement Computing PC104-DAS16JR/16 Analog Input (A/D)

---

**Purpose** PC104-DAS16JR/16 Analog Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of different range for each channel.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

This choice must correspond to the MUX-switch setting on the board.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Measurement Computing PC104-DAS16JR/16 Analog Input (A/D)

---

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing PC104-DAS16JR/16 Digital Input

**Purpose** PC104-DAS16JR/16 Digital Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

Enter a number between 1 and 4 to select the number of digital input lines used. This driver does not allow the selection of individual digital input lines.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing PC104-DAS16JR/16 Digital Output

---

**Purpose** PC104-DAS16JR/16 Digital Output

**Library** xPC Target Library for Measurement Computing

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Number of Channels**

Enter a number between 1 and 4 to select the number of digital output lines used. This driver does not allow the selection of individual digital output lines.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing PC104-DIO48

---

**Board** Measurement Computing PC104-DIO48

**General Description** The PC104-DIO48 is an I/O board with 48 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing PC104-DIO48 Digital Input
- Measurement Computing PC104-DIO48 Digital Output

<b>Board Characteristics</b>	Board name	PC104-DIO48
	Manufacturer	Measurement Computing
	Bus type	PC/104
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# Measurement Computing PC104-DIO48 Digital Input

---

**Purpose** PC104-DIO48 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The CIO-DIO48 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.



# Measurement Computing PC104-DIO48 Digital Input

---

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **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

# Measurement Computing PC104-DIO48 Digital Output

**Purpose** PC104-DIO48 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The PC104-DIO48 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs. Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **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

# Measurement Computing PCI-CTR05

---

**Board** Measurement Computing PCI-CTR05

**General Description** The PCI-CTR05 is an I/O board with 5 counter/timer channels (16-bit). It contains one AM9513A counter/timer chip. For additional information about the various counter/timer modes of that chip see the AM9513A data sheet which is part of the board documentation.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-CTR05 Counter PWM
- Measurement Computing PCI-CTR05 Counter PWM & ARM
- Measurement Computing PCI-CTR05 Counter FM
- Measurement Computing PCI-CTR05 Counter FM & ARM
- Measurement Computing PCI-CTR05 PWM Capture
- Measurement Computing PCI-CTR05 Frequency Capture
- Measurement Computing PCI-CTRxx

<b>Board Characteristics</b>	Board name	PCI-CTR05
	Manufacturer	Measurement Computing
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing PCI-CTR05 Counter PWM

**Purpose** PCI-CTR05 Counter PWM block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-CTR05 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 which defines the variable duty cycle between 0 and 1. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### **Relative output frequency**

Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 17.5 kHz, then choose F2=100kHz as the **Frequency base** and enter 0.175 as the **Relative output frequency**.  $100 \text{ kHz} \times 0.175 = 17.5 \text{ kHz}$

# Measurement Computing PCI-CTR05 Counter PWM

---

## 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 gets 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)

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-CTR05 Counter PWM & ARM

**Purpose** PCI-CTR05 Counter PWM & ARM block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-CTR05 has one AM9513A chip with 5 counters.

The PCI-CTR05 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 to arm and disarm the counter by 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

TTL

### Block Input Data Type

Duty cycle: Double  
Arm: Double

### Scaling

0 to 1  
<0.5 disarmed  
≥0.5 armed

## 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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### Relative output frequency

Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 17.5 kHz, then choose F2=100kHz as the **Frequency base** and enter 0.175 as the **Relative output frequency**.  $100 \text{ kHz} \times 0.175 = 17.5 \text{ kHz}$

# Measurement Computing PCI-CTR05 Counter PWM & ARM

---

## 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)

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# Measurement Computing PCI-CTR05 Counter FM

---

**Purpose** PCI-CTR05 Counter FM block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-CTR05 has one AM9513A chip with 5 counters.  
The PCI-CTR05 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 PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### **Output duty cycle**

Enter a value between 0 and 1 to set the duty cycle of the square wave. The Duty Cycle is held fixed during execution of the target application.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

# Measurement Computing PCI-CTR05 Counter FM

---

## 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 gets 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)

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-CTR05 Counter FM & ARM

---

**Purpose** PCI-CTR05 Counter FM & ARM block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-CTR05 has one AM9513A chip with 5 counters.

The PCI-CTR05 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 to arm and disarm the counter by 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	Variable frequency: Double Arm: Double	<0.5 disarmed ≥0.5 armed

## 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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### Output duty cycle

Enter a value between 0 and 1 to set the duty cycle of the square wave. The Duty Cycle is held fixed during execution of the target application.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

## Level sequence of square wave

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the **Output duty cycle** defined in the setting above define 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)

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-CTR05 PWM Capture

**Purpose** PCI-CTR05 PWM Capture block

**Library** xPC Target Library for Measurement Computing

**Note** This block programs the AMD9513A for capturing PWM signals by using two counters. One counter measures the cycle duration, and the other counter measures the duration 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 has to enter the pins named GATE of both corresponding counter channels (parallel wiring). Both CLK pins have to be left unconnected.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	0 to 1

## Block Parameters

### Counter

From the list, choose 1&2, 2&3, 3&4, 4&5. This selects which two counters the driver block uses to determine the PWM. In each case, one block is needed for each counter.

### Frequency base

From the list, choose F1=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### Sample time

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

# Measurement Computing PCI-CTR05 PWM Capture

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-CTR05 Frequency Capture

**Purpose** PCI-CTR05 Frequency Capture block

**Library** xPC Target Library for Measurement Computing

**Note** This block programs the AMD9513A 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 has to enter the pin named GATE of the corresponding counter channel. The CLK pin has to be left unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## Block Parameters

### Counter

From the list, choose 1, 2, 3, 4 or 5. 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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### Sample time

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# Measurement Computing PCI-CTR05 Frequency Capture

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



**Purpose** PCI-CTRxx block

**Library** xPC Target Library for Measurement Computing

**Note** You can use this block to program the AMD9513A counter. The PWM, PWM & ARM, FM, FM & ARM, PWM Capture, and Frequency Capture blocks use this block in their underlying subsystems. The API for this block is not currently documented.

# Measurement Computing PCI-DAS1200

---

## Board

Measurement Computing PCI-DAS1200

## General Description

The PCI-DAS1200 is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 330 kHz, 2 analog output (D/A) channels (12-bit), and 24 digital input and output lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DAS1200 Analog Input (A/D)
- Measurement Computing PCI-DAS1200 Analog Output (D/A)
- Measurement Computing PCI-DAS1200 Digital Input
- Measurement Computing PCI-DAS1200 Digital Output

## Board Characteristics

Board name	PCI-DAS1200
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	A/D: No, D/A: Yes, Digital I/O: Yes
Multiple board support	Yes

# Measurement Computing PCI-DAS1200 Analog Input (A/D)

---

**Purpose** PCI-DAS1200 Analog Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of different range for each channel.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

This choice must correspond to the MUX-switch setting on the board.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Measurement Computing PCI-DAS1200 Analog Input (A/D)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1200 Analog Output (D/A)

**Purpose** PCI-DAS1200 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## Block Parameters

### Channel vector

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Measurement Computing PCI-DAS1200 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be placed according to the ranges used.

## **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**

Base sample time of a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# Measurement Computing PCI-DAS1200 Analog Output (D/A)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1200 Digital Input

---

**Purpose** PCI-DAS1200 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DAS1200 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.



# Measurement Computing PCI-DAS1200 Digital Input

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1200 Digital Output

**Purpose** PCI-DAS1200 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DAS1200 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	$< 0.5 = \text{TTL low}$ $\geq 0.5 = \text{TTL high}$

## **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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1200/JR

---

## Board

Measurement Computing PCI-DAS1200/JR

## General Block Description

The PCI-DAS1200/JR is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 330 kHz, and 24 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DAS1200/JR Analog Input (A/D)
- Measurement Computing PCI-DAS1200/JR Digital Input
- Measurement Computing PCI-DAS1200/JR Digital Output

## Board Characteristics

Board name	PCI-DAS1200/JR
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	A/D: No, Digital I/O: Yes
Multiple board support	Yes

# Measurement Computing PCI-DAS1200/JR Analog Input (A/D)

---

**Purpose** PCI-DAS1200/JR Analog Input block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of different range for each channel.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

This choice must correspond to the MUX-switch setting on the board.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Measurement Computing PCI-DAS1200/JR Analog Input (A/D)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1200/JR Digital Input

**Purpose** PCI-DAS1200/JR Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DAS1200/JR has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

### Hardware Input

TTL

### Block Output Data Type

Double

### Scaling

TTL low = 0.0

TTL high = 1.0

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing PCI-DAS1200/JR Digital Input

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# Measurement Computing PCI-DAS1200/JR Digital Output

**Purpose** PCI-DAS1200/JR Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DAS1200 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DAS1200/JR Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1602/12

---

## Board

Measurement Computing PCI-DAS1602/12

## General Description

The PCI-DAS1602/12 is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sampling rate of 200 kHz, 2 analog output (D/A) channels (12-bit), and 24 digital input and output lines and 3 counters (16-bit).

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DAS1602/12 Analog Input (A/D)
- Measurement Computing PCI-DAS1602/12 Analog Output (D/A)
- Measurement Computing PCI-DAS 1602/12 Digital Input
- Measurement Computing PCI-DAS1602/12 Digital Output

---

**Note** xPC Target supports only 24 digital I/O lines and does not support the counters on this board.

---

## Board Characteristics

Board Name	PCI-DAS1602/12
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCI-DAS1602/12 Analog Input (A/D)

---

**Purpose** PCI-DAS1602/12 Analog Input block

**Library** xPC Target Library for Measurement Computing

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Number of channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of a different range for each channel.

If a bipolar range is used, the bipolar switch on the board must be in the bipolar position. if a unipolar range is used, the bipolar switch must be in the unipolar position.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

# Measurement Computing PCI-DAS1602/12 Analog Input (A/D)

---

This choice must correspond to the MUX-switch setting on the board.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1602/12 Analog Output (D/A)

---

**Purpose** PCI-DAS1602/12 Analog Output block

**Library** xPC Target Library for Measurement Computing

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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 channels begin with 1 even if the board manufacturer starts numbering channels with 0.

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Measurement Computing PCI-DAS1602/12 Analog Output (D/A)

---

For example, if the first channel is -10 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**

Base sample time of a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS 1602/12 Digital Input

**Purpose** PCI-DAS 1602/12 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/12 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling of Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.



# Measurement Computing PCI-DAS 1602/12 Digital Input

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1602/12 Digital Output

---

**Purpose** PCI-DAS1602/12 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/12 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1602/16

---

## Board

Measurement Computing PCI-DAS1602/16

## General Description

The PCI-DAS1602/16 is an I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit) with a maximum sampling rate of 200 kHz, 2 analog output (D/A) channels (16-bit), and 24 digital input and output lines and 3 counters (16-bit).

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DAS1602/16 Analog Input (A/D)
- Measurement Computing PCI-DAS1602/16 Analog Output (D/A)
- Measurement Computing PCI-DAS 1602/16 Digital Input
- Measurement Computing PCI-DAS1602/16 Digital Output

---

**Note** xPC Target supports only 24 digital I/O lines and does not support the counters on this board.

---

## Board Characteristics

Board Name	PCI-DAS1602/16
Manufacturer	Measurement Computing
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCI-DAS1602/16 Analog Input (A/D)

---

**Purpose** PCI-DAS1602/16 Analog Input block

**Library** xPC Target Library for Measurement Computing

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Number of channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of a different range for each channel.

If a bipolar range is used, the bipolar switch on the board must be in the bipolar position. If a unipolar range is used, the bipolar switch must be in the unipolar position.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

# Measurement Computing PCI-DAS1602/16 Analog Input (A/D)

---

This choice must correspond to the MUX-switch setting on the board.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1602/16 Analog Output (D/A)

**Purpose** PCI-DAS1602/16 Analog Output block

**Library** xPC Target Library for Measurement Computing

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers 1 or 2. 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 channels begin with 1 even if the board manufacturer starts numbering channels with 0.

### **Range vector**

Enter a vector or range codes to specify the ranges of each of the channels. This must be a vector of the same length as the channel vector. Otherwise, a scalar is assumed to apply to each channel.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Measurement Computing PCI-DAS1602/16 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be planted according to the ranges used.

## **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**

Base sample time of a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.



# Measurement Computing PCI-DAS1602/16 Analog Output (D/A)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpci
```

# Measurement Computing PCI-DAS 1602/16 Digital Input

**Purpose** PCI-DAS 1602/16 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/16 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling of Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

**Sample time**

Enter a base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAS1602/16 Digital Output

---

**Purpose** PCI-DAS1602/16 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/16 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling of Input to Output

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

## 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 of 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 a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA02/12

---

## Board

Measurement Computing PCI-DDA02/12

## General Description

The PCI-DDA02/12) is an I/O board with 2 analog output (D/A) channels (12-bit), and 48 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DDA02/12 Analog Output (D/A)
- Measurement Computing PCI-DDA02/12 Digital Input
- Measurement Computing PCI-DDA02/12 Digital Output

## Board Characteristics

Board name	PCI-DDA02/12
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCI-DDA02/12 Analog Output (D/A)

**Purpose** PCI-DDA02/12 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## Block Parameters

### Channel vector

Enter numbers 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +10	10

# Measurement Computing PCI-DDA02/12 Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be placed according to the ranges used.

## **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**

Base sample time of a multiple of the base sample time.



# Measurement Computing PCI-DDA02/12 Analog Output (D/A)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

getxpcpci

# Measurement Computing PCI-DDA02/12 Digital Input

**Purpose** PCI-DDA02/12 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA02/12 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DDA02/12 Digital Input

---

depending on which driver block is chosen. In each case, one block is needed for each port.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA02/12 Digital Output

**Purpose** PCI-DDA02/12 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA02/12 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA02/16

---

**Board** Measurement Computing PCI-DDA02/16

**General Description** The PCI-DDA02/16) is an I/O board with 2 analog output (D/A) channels (12-bit), and 48 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DDA02/16 Analog Output (D/A)
- Measurement Computing PCI-DDA02/16 Digital Input
- Measurement Computing PCI-DDA02/16 Digital Output

<b>Board Characteristics</b>	Board name	PCI-DDA06/12
	Manufacturer	Measurement Computing
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing PCI-DDA02/16 Analog Output (D/A)

**Purpose** PCI-DDA02/16 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +10	10

# Measurement Computing PCI-DDA02/16 Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be placed according to the ranges used.

## 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

Base sample time of a multiple of the base sample time.



# Measurement Computing PCI-DDA02/16 Analog Output (D/A)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA02/16 Digital Input

**Purpose** PCI-DDA02/16 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA02/16 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DDA02/16 Digital Input

---

depending on which driver block is chosen. In each case, one block is needed for each port.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA02/16 Digital Output

**Purpose** PCI-DDA02/16 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA02/16 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	$< 0.5 = \text{TTL low}$ $\geq 0.5 = \text{TTL high}$

## **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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA04/12

---

## Board

Measurement Computing PCI-DDA04/12

## General Description

The PCI-DDA04/12) is an I/O board with 4 analog output (D/A) channels (12-bit), and 48 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DDA04/12 Analog Output (D/A)
- Measurement Computing PCI-DDA04/12 Digital Input
- Measurement Computing PCI-DDA04/12 Digital Output

## Board Characteristics

Board name	PCI-DDA04/12
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCI-DDA04/12 Analog Output (D/A)

**Purpose** PCI-DDA04/12 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## 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 used. For example, to use the first and second analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +10	10

# Measurement Computing PCI-DDA04/12 Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be placed according to the ranges used.

## 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

Base sample time of a multiple of the base sample time.



# Measurement Computing PCI-DDA04/12 Analog Output (D/A)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA04/12 Digital Input

**Purpose** PCI-DDA04/12 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA4/12 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing PCI-DDA04/12 Digital Input

---

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA04/12 Digital Output

**Purpose** PCI-DDA04/12 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA04/12 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA04/16

---

## Board

Measurement Computing PCI-DDA04/16

## General Description

The PCI-DDA04/16) is an I/O board with 4 analog output (D/A) channels (16-bit), and 48 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DDA04/16 Analog Output (D/A)
- Measurement Computing PCI-DDA04/16 Digital Input
- Measurement Computing PCI-DDA04/16 Digital Output

## Board Characteristics

Board name	PCI-DDA04/16
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCI-DDA04/16 Analog Output (D/A)

**Purpose** PCI-DDA04/16 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## 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 used. For example, to use the first and second analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +10	10

# Measurement Computing PCI-DDA04/16 Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be placed according to the ranges used.

## **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**

Base sample time of a multiple of the base sample time.



# Measurement Computing PCI-DDA04/16 Analog Output (D/A)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA04/16 Digital Input

**Purpose** PCI-DDA04/16 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA4/16 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DDA04/16 Digital Input

---

depending on which driver block is chosen. In each case, one block is needed for each port.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA04/16 Digital Output

**Purpose** PCI-DDA04/16 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA04/16 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA08/12

---

**Board** Measurement Computing PCI-DDA08/12

**General Description** The PCI-DDA08/12) is an I/O board with 8 analog output (A/D) channels (16-bit), and 48 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DDA08/12 Analog Output (D/A)
- Measurement Computing PCI-DDA08/12 Digital Input
- Measurement Computing PCI-DDA08/12 Digital Output

<b>Board Characteristics</b>	Board name	PCI-DDA08/12
	Manufacturer	Measurement Computing
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing PCI-DDA08/12 Analog Output (D/A)

**Purpose** PCI-DDA08/12 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 8. 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 channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +10	10

# Measurement Computing PCI-DDA08/12 Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be placed according to the ranges used.

## 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

Base sample time of a multiple of the base sample time.



# Measurement Computing PCI-DDA08/12 Analog Output (D/A)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA08/12 Digital Input

**Purpose** PCI-DDA08/12 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA08/12 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing PCI-DDA08/12 Digital Input

---

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA08/12 Digital Output

**Purpose** PCI-DDA08/12 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA08/12 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA08/16

---

## Board

Measurement Computing PCI-DDA08/16

## General Description

The PCI-DDA08/16) is an I/O board with 8 analog output (A/D) channels (12-bit), and 48 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DDA08/16 Analog Output (D/A)
- Measurement Computing PCI-DDA08/16 Digital Input
- Measurement Computing PCI-DDA08/16 Digital Output

## Board Characteristics

Board name	PCI-DDA06/12
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCI-DDA08/16 Analog Output (D/A)

**Purpose** PCI-DDA08/16 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 8. 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 channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +10	10

# Measurement Computing PCI-DDA08/16 Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +5	5
0 to +2.5	2.5

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be placed according to the ranges used.

## 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

Base sample time of a multiple of the base sample time.



# Measurement Computing PCI-DDA08/16 Analog Output (D/A)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

getxpcpci

# Measurement Computing PCI-DDA08/16 Digital Input

**Purpose** PCI-DDA08/16 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA08/16 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DDA08/16 Digital Input

---

depending on which driver block is chosen. In each case, one block is needed for each port.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DDA08/16 Digital Output

**Purpose** PCI-DDA08/16 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DDA08/16 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DIO24

---

**Board** Measurement Computing PCI-DIO24

**General Description** The PCI-DIO24 is an I/O board with 24 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DIO24 Digital Input
- Measurement Computing PCI-DIO24 Digital Output
- Measurement Computing PCI-DIO24 Signal Conditioning

<b>Board Characteristics</b>	Board name	PCI-DIO24
	Manufacturer	Measurement Computing
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing PCI-DIO24 Digital Input

**Purpose** PCI-DIO24 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO24 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 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.

# Measurement Computing PCI-DIO24 Digital Input

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# Measurement Computing PCI-DIO24 Digital Output

**Purpose** PCI-DIO24 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO24 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DIO24 Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DIO24 Signal Conditioning

---

**Purpose** PCI-DIO24 Signal Conditioning

**Library** xPC Target Library for Measurement Computing

**Block Parameters** **Genix initialization file (path\file)**  
Provide the filename of the Genix initialization file.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

getxpcpci

# Measurement Computing PCI-DIO24H

---

**Board** Measurement Computing PCI-DIO24H

**General Description** The PCI-DIO24H is an I/O board with 24 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DIO24H Digital Input
- Measurement Computing PCI-DIO24H Digital Output

<b>Board Characteristics</b>	Board name	PCI-DIO24H
	Manufacturer	Measurement Computing
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing PCI-DIO24H Digital Input

**Purpose** PCI-DIO24H Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO24H has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

### Hardware Input

TTL

### Block Output Data Type

Double

### Scaling

TTL low = 0.0  
TTL high = 1.0

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing PCI-DIO24H Digital Input

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DIO24H Digital Output

**Purpose** PCI-DIO24H Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO24H has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DIO24H Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



**Board** Measurement Computing PCI-DIO48H

**General Description** The PCI-DIO48H is an I/O board with 48 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DIO48H Digital Input
- Measurement Computing PCI-DIO48H Digital Output

<b>Board Characteristics</b>	Board name	PCI-DIO48H
	Manufacturer	Measurement Computing
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing PCI-DIO48H Digital Input

---

**Purpose** PCI-DIO48H Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO48H has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing PCI-DIO48H Digital Input

---

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DIO48H Digital Output

**Purpose** PCI-DIO48H Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO48H has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DIO48H Digital Output

---

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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DIO96H

---

**Board** Measurement Computing PCI-DIO96H

**General Description** The PCI-DIO96H is an I/O board with 96 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DIO96H Digital Input
- Measurement Computing PCI-DIO96H Digital Output

<b>Board Characteristics</b>	Board name	PCI-DIO96H
	Manufacturer	Measurement Computing
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing PCI-DIO96H Digital Input

**Purpose** PCI-DIO96H Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO96H has four 8255 chips (1,2,3,4). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing PCI-DIO96H Digital Input

---

## **Chip**

From the list choose 1, 2, 3, or 4. The **Chip** parameter defines which of the four 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# Measurement Computing PCI-DIO96H Digital Output

**Purpose** PCI-DIO96H Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO96H has four 8255 chips (1,2,3,4). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing PCI-DIO96H Digital Output

---

## **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.

## **Chip**

From the list choose 1, 2, 3, or 4. The **Chip** parameter defines which of the four 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** Measurement Computing PCI-DIO96

**General Description** The PCI-DIO96 is an I/O board with 96 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DIO96 Digital Input
- Measurement Computing PCI-DIO96 Digital Output

<b>Board Characteristics</b>	Board name	PCI-DIO96
	Manufacturer	Measurement Computing
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing PCI-DIO96 Digital Input

---

**Purpose** PCI-DIO96 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO96 has four 8255 chips (1,2,3,4). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The **Port** parameter defines which port of the current 8255 chip is used for this driver block.

### **Chip**

From the list choose 1, 2, 3, or 4. The **Chip** parameter defines which of the four 8255 chips is used.

# Measurement Computing PCI-DIO96 Digital Input

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DIO96 Digital Output

---

**Purpose** PCI-DIO96 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DIO96 has four 8255 chips (1,2,3,4). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	$< 0.5 = \text{TTL low}$ $\geq 0.5 = \text{TTL high}$

## **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 of 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 either A, B, or C. The **Port** parameter defines which port of the current 8255 chip is used for this driver block.

### **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.

## **Chip**

From the list choose 1, 2, 3, or 4. The **Chip** parameter defines which of the four 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-PDISO8

---

## Board

Measurement Computing PCI-PDISO8

## General Description

The PCI-PDISO8 is an I/O board with eight inputs and eight relay outputs.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-PDISO8 Digital Input
- Measurement Computing PCI-PDISO8 Digital Output

## Board Characteristics

Board name	PCI-PDISO8
Manufacturer	Measurement Computing
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Yes
Multiple board support	Yes



# Measurement Computing PCI-PDISO8 Digital Input

**Purpose** PCI-PDISO8 Digital Input block

**Library** xPC Target Library for Measurement Computing

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	OPTO	Double	Low (opto off) = 0.0 High (opto on) = 1.0

The input is rectified before being applied to the opto isolator on the input. With no additional input current limiting resistor, the opto turns on when  $|V_{in}| > 3$  volts.

## 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 of the digital inputs, enter

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

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

### Filter vector

This vector controls the 5 ms noise filter. It is the same length as the channel vector. A value of 1 turns the filter on, and a value of 0 turns the filter off. If you specify a scalar, this value is applied to all channels.

### Port

From the list choose A because this board only has eight channels.

# Measurement Computing PCI-PDISO8 Digital Input

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-PDISO8 Digital Output

**Purpose** PCI-PDISO8 Digital Output

**Library** xPC Target Library for Measurement Computing

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	RELAY	Double	< 0.5 = relay off ≥ 0.5 = relay on

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital output lines to drive. 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 of the digital outputs, enter

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

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

### **Port**

From the list choose A because this board only has eight channels.

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# Measurement Computing PCI-PDISO8 Digital Output

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** Measurement Computing PCI-PDISO16

**General Description**

The PCI-PDISO16 is an I/O board with 16 inputs and 16 relay outputs. xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-PDISO16 Digital Input
- Measurement Computing PCI-PDISO16 Digital Output

**Board Characteristics**

Board name	PCI-PDISO16
Manufacturer	Measurement Computing
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCI-PDISO16 Digital Input

**Purpose** PCI-PDISO16 Digital Input block

**Library** xPC Target Library for Measurement Computing

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	OPTO	Double	Low (opto off) = 0.0 High (opto on) = 1.0

The input is rectified before being applied to the opto isolator on the input. With no additional input current limiting resistor, the opto turns on when  $|V_{in}| > 3$  volts.

## 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 of the digital inputs, enter

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

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

### Filter vector

This vector controls the 5 ms noise filter. It is the same length as the channel vector. A value of 1 turns the filter on, and a value of 0 turns the filter off. If you specify a scalar, this value is applied to all channels.

### Port

From the list choose either A or B. The **Port** parameter defines which port is used for this driver block. Each port has a maximum

# Measurement Computing PCI-PDISO16 Digital Input

---

of 8 digital inputs. More than one block can be connected to the same port, but each channel can only be referenced by one block at a time.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-PDISO16 Digital Output

---

**Purpose** PCI-PDISO16 Digital Output block

**Library** xPC Target Library for Measurement Computing

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	RELAY	Double	< 0.5 = relay off ≥ 0.5 = relay on

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital output lines to drive. 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 of the digital outputs, enter

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

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

### **Port**

From the list choose either A or B. The **Port** parameter defines which port is used for this driver block. Each port has a maximum of 8 digital outputs. More than one output block can be used with the same port, but each output channel can only be controlled by one block at a time.

### **Sample time**

Enter a base sample time or a multiple of the base sample time.



# Measurement Computing PCI-PDISO16 Digital Output

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCIM-DAS1602/16

---

## Board

Measurement Computing PCIM-DAS1602/16

## General Description

The PCIM-DAS1602/16 is an I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit) with a maximum sampling rate of 100 kHz, 2 analog output (D/A) channels (16-bit), and 24 digital input and output lines and 3 counters (16-bit).

xPC Target supports this board with these driver blocks:

- Measurement Computing PCIM-DAS1602/16 Analog Input (A/D)
- Measurement Computing PCIM-DAS1602/16 Analog Output (D/A)
- Measurement Computing PCIM-DAS 1602/16 Digital Input
- Measurement Computing PCIM-DAS1602/16 Digital Output

---

**Note** xPC Target supports only 24 digital I/O lines and does not support the counters on this board.

---

## Board Characteristics

Board Name	PCIM-DAS1602/16
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCIM-DAS1602/16 Analog Input (A/D)

**Purpose** PCIM-DAS1602/16 Analog Input block

**Library** xPC Target Library for Measurement Computing

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Number of channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or to mix single-ended and differential inputs.

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

### **Range**

From the list, choose either +-10V (-10 volts to +10 volts), +-5V, +-2.5V, +-1.25V, +-0.625V, 0-10V, 0-5V, 0-2.5V, or 0-1.25V. This driver does not allow the selection of a different range for each channel.

If a bipolar range is used, the bipolar switch on the board must be in the bipolar position. If a unipolar range is used, the bipolar switch must be in the unipolar position.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

# Measurement Computing PCIM-DAS1602/16 Analog Input (A/D)

---

This choice must correspond to the MUX-switch setting on the board.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCIM-DAS1602/16 Analog Output (D/A)

**Purpose** PCIM-DAS1602/16 Analog Output block

**Library** xPC Target Library for Measurement Computing

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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 channels begin with 1 even if the board manufacturer starts numbering channels with 0.

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Measurement Computing PCIM-DAS1602/16 Analog Output (D/A)

---

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board. Also the bipolar/unipolar jumpers must be inserted according to the ranges used.

## **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**

Base sample time of a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# Measurement Computing PCIM-DAS1602/16 Analog Output (D/A)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCIM-DAS 1602/16 Digital Input

---

**Purpose** PCIM-DAS 1602/16 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/16 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling of Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.



**Sample time**

Enter a base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCIM-DAS1602/16 Digital Output

---

**Purpose** PCIM-DAS1602/16 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCIM-DAS1601/16 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The port name defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCIM-DDA06/16

---

## Board

Measurement Computing PCIM-DDA06/16

## General Description

The PCIM-DDA06/16 is an I/O board with 6 analog output (D/A) channels (16-bit), and 24 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Measurement Computing PCIM-DDA06/16 Analog Output (D/A)
- Measurement Computing PCIM-DDA06/16 Digital Input
- Measurement Computing PCIM-DDA06/16 Digital Output

## Board Characteristics

Board name	PCIM-DDA06/16
Manufacturer	Measurement Computing
Bus type	PCIM
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCIM-DDA06/16 Analog Output (D/A)

**Purpose** PCIM-DDA06/16 Analog Output block

**Library** xPC Target Library for Measurement Computing

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 6. 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 channels, enter

[1,2]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
-1.67 to +1.67	-1.67
-0.625 to +0.625	-0.625

# Measurement Computing PCIM-DDA06/16 Analog Output (D/A)

---

Input Range (V)	Range Code
0 to +10	10
0 to +5	5
0 to +2.5	2.5
0 to +1.67	1.67

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the DIP switch settings on the board.

---

**Note** For xPC Target, set the Simultaneous Transfer jumper to the XFER setting.

---

## 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.

# Measurement Computing PCIM-DDA06/16 Analog Output (D/A)

---

## **Sample time**

Base sample time of a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCIM-DDA06/16 Digital Input

**Purpose** PCIM-DDA06/16 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCIM-DDA6/16 has a 8255 chip with three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

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

## **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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 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.

**Sample time**

Enter a base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

getxpcpci

# Measurement Computing PCIM-DDA06/16 Digital Output

---

**Purpose** PCIM-DDA06/16 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCIM-DDA06/16 has a 8255 chip with three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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 two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DUAL-AC5

---

**Board** Measurement Computing PCI-DUAL-AC5

**General Description** The PCI-DUAL-AC5 is an I/O board with 48 digital I/O lines. xPC Target supports this board with these driver blocks:

- Measurement Computing PCI-DUAL-AC5 Digital Input
- Measurement Computing PCI-DUAL-AC5 Digital Output

<b>Board Characteristics</b>	Board name	PCI-DUAL-AC5
	Manufacturer	Measurement Computing
	Bus type	compact PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Measurement Computing PCI-DUAL-AC5 Digital Input

**Purpose** PCI-DUAL-AC5 Digital Input

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DUAL-AC5 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# Measurement Computing PCI-DUAL-AC5 Digital Input

---

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DUAL-AC5 Digital Output

**Purpose** PCI-DUAL-AC5 Digital Output

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DUAL-AC5 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 either A, B, or C. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DUAL-AC5 Digital Output

---

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.

## **Chip**

From the list choose 1 or 2. The **Chip** parameter defines which of the two 8255 chips is used.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



## Board

Measurement Computing PCI-QUAD04

## General Description

The PCI-QUAD04 is a 24-bit counting board with 4 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.

xPC Target supports this board with this driver block:

- Measurement Computing PCI-QUAD04 Incremental Encoder

See Measurement Computing PCI-QUAD04 Incremental Encoder if you have an earlier version of this driver.

Note on usage

There is a hardware limitation with the PCI-Quad04 boards in the way that they handle unconnected inputs. Even with the **Index input resets counter** check box disabled, noise on the index input line might cause the counter chip on the board to latch an incorrect value. This value differs from the correct value by a multiple of 256. The frequency of occurrence of the bad reads depends on the count frequency coming from the encoder. High count rates result in more bad reads.

To work around this issue, try the following:

- 1** If the index signal from an encoder is not being used, connect the index inputs.
- 2** Connect the index+ input to +5 volts.
- 3** Connect the index- input to ground.

These connections should be made as close to the board as possible to minimize any crosstalk at high count frequencies.

# Measurement Computing PCI-QUAD04

---

## **Board Characteristics**

Board name	PCI-QUAD04
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Measurement Computing PCI-QUAD04 Incremental Encoder

---

**Purpose** PCI-QUAD04 Incremental Encoder

**Library** xPC Target Library for Measurement Computing

**Description** This driver block has one block output: **Count**.

The raw count value is a 24-bit integer that is converted to a double precision float output from the block. The 24-bit value can be interpreted in different ways (see **Count Range**).

In this section, the clockwise direction is the direction in which the counter increments. The counterclockwise direction is the direction in which the counter decrements. Note that the counter direction depends on the encoder manufacturer and the way in which the encoder is installed.

---

**Note** If you are using an earlier version of this driver block, see the description at the end of this section.

---

## Block Parameters

**Channel**  
From the list, choose 1, 2, 3, or 4. This parameter specifies the channel you use for this block. For the same board (same PCI slot) two blocks cannot have the same channel number.

**Index input resets counter**  
Choose this check box to control the behavior when the index is reached. For example, a rotary encoder typically reaches the index at one particular position in its range of motion. Setting this check box to reset the count value allows you to get a repeatable position from an encoder. When the encoder reaches the index (asserts the index), the counter value is set to the value given in the **Count Limit or Reset Value** field.

# Measurement Computing PCI-QUAD04 Incremental Encoder

## Index Polarity

From the list, select the index pulse slope that the board should detect. Select Positive Index for a positive slope; select Negative Index for a negative slope.

## Counting Mode

From the list, select a counting mode (Normal, Range Limit, Non-recycle, or Modulo-N). The output of the block depends on both the setting of **Index input resets counter** and on the counting mode as described in the following table.

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Normal	<p>Upon model start, the output count is initialized to <b>Count Limit or Reset Value</b>. The output count increments or decreases one full revolution from the <b>Count Limit or Reset Value</b>. The output count is reset to this value each time the index is reached.</p> <p>For example, if <b>Index input resets counter</b> is selected, and a rotary encoder has 1024 counts per revolution, and a <b>Count Limit or Reset Value</b> value of 5000, a full clockwise revolution of the encoder increments the counter to 6023, at which the index is reached. The counter then resets to 5000.</p>	<p>Upon model start, the counter is initialized to 0. The counter then increments or decrements until it either overflows or underflows. For a rotary encoder, the number of revolutions is the number of output counts divided by the number of counts per revolution. The resulting integer is the number of revolutions; the fractional part can be converted to an angle.</p>

# Measurement Computing PCI-QUAD04 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Range Limit	<p>Upon model start, the counter increments or decrements until it reaches the count limit <b>Count Limit or Reset Value</b>. If the clock motion is clockwise, the counter stays at <b>Count Limit or Reset Value</b> once it reaches that value. If the clock motion is counterclockwise, the counter decrements one revolution then resets to <b>Count Limit or Reset Value</b>.</p>	<p>Upon model start, the counter is initialized to 0. In the clockwise direction, the counter increments until it saturates at <b>Count Limit or Reset Value</b>. In the counterclockwise direction, the counter decrements until it saturates at 0.</p> <p>You can use this setting to set a repeatable 0 when there is a mechanical stop. At model start, the count is initialized to 0. The machine then shifts to the negative direction stop. At that point, the encoder reports 0 at the stop and all other positions will be positive and repeatable.</p>

# Measurement Computing PCI-QUAD04 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
Non-recycle	<p>Upon model start, the counter is initialized to <b>Count Limit or Reset Value</b>. With counterclockwise rotation, the counter decrements and saturates at 0. If the encoder reaches the index before the counter saturates at 0, the counter resets to <b>Count Limit or Reset Value</b>. With clockwise rotation, the counter increments until the encoder reaches the index. When the encoder reaches the index, the counter resets to <b>Count Limit or Reset Value</b>. You can use this setting to place 0 at the first index inside a mechanical limit.</p>	<p>The behavior of this option is undefined.</p>
Modulo-N	<p>The behavior of this option is undefined.</p>	<p>Upon model start, the counter is initialized to 0. In the clockwise direction, the counter increments to <b>Count Limit or Reset Value</b>, then resets to 0 and repeats. When the counter decrements, it counts to 0, then resets to the <b>Count Limit or Reset Value</b> value and continues to decrement. You can use this setting to repeat intervals that do not match the distance between indexes (resulting, for example, from gear ratios).</p>

## Count Limit or Reset Value

The preset register (PR) on the counter chip is a 24-bit quantity. The counter uses this value as a limit for the **Range Limit**, **Non-recycle**, and **Modulo-N** counting modes.

When **Index input resets counter** is selected, this value acts as both the initial value at model start and the value that an index resets the count to.

You can enter negative limit values, but the counter treats them as large positive values. For example, a limit of -1 results in the **Modulo-N** range from 0 to  $2^{24} - 1$ , not from -1 to 0.

## Count Range

From the list, choose  $0 \dots 2^{24} - 1$  or  $-2^{23} \dots +2^{23} - 1$ . The counter uses a 24-bit accumulator which overflows from  $2^{24} - 1$  to 0. This same overflow can be interpreted as the transition from -1 to 0. Selecting the unipolar range 0 to  $2^{24} - 1$  puts the discontinuity in counts at 0. Selecting the bipolar range of  $(-2)^{23}$  to  $2^{23} - 1$  puts the discontinuity far away from 0. The bipolar range is achieved by sign extending the 24-bit count to 32 bits before converting to double.

## Count Speed

From the list, choose non-quadrature, 1X, 2X, or 4X:

- non-quadrature — Counts the rising edges of the A input, losing direction information in the process. The presence of a quadrature signal on the B input might cause undefined results. You should disconnect the B and index inputs.
- 1X — Counts up or down once per complete cycle of the quadrature signal.
- 2X — Counts up or down twice per complete cycle of the quadrature signal.
- 4X — Counts up or down four times per complete cycle of the quadrature signal.

# Measurement Computing PCI-QUAD04 Incremental Encoder

---

## **Filter prescale factor**

Enter a base prescale factor for digital filtering. This filter helps eliminate high frequency noise. Enter a value from 0 to ff in hexadecimal (0 to 256 in decimal). For example, for a prescale factor value of ff in hexadecimal, enter

```
0xff
```

## **Sample time**

Enter a 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). Enter -1 to inherit the model sample time from another block. The sample time is in seconds.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



**Board** Measurement Computing PCI-DAS-TC

**General Description** The PCI-DAS-TC is an I/O board with 16 differential analog thermocouple input channels. The thermocouple signals are converted by a high frequency synchronous V-F A/D converter. The board is equipped with its own micro-controller responsible for calibration, cold junction compensation, moving average calculations, conversion, and conversion into physical engineering units. The on-board micro-controller significantly off loads the main CPU.

xPC Target supports this board with this driver block:

- Measurement Computing PCI-DAS-TC Thermocouple

<b>Board Characteristics</b>	Board name	PCI-DAS-TC
	Manufacturer	Measurement Computing
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# Measurement Computing PCI-DAS-TC Thermocouple

---

**Purpose** PCI-DAS-TC Thermocouple

**Library** xPC Target Library for Measurement Computing

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	temperature in either degrees C, K, or F.

## **Block Parameters**

### **Conversion rate (interrogation time)**

From the list, choose either 50Hz, 60Hz, or 400 Hz. This is the conversion rate for the V-F A/D converter. The conversion rate is the same for all input channels.

### **Number of samples for moving average**

From the list, choose a value from **1** to **16**. Converted signal values are put into a cyclic buffer of size **n** which is used to calculate the moving average over these **n** samples.

### **Number of channels to be acquired (1..n)**

From the list, choose a value from **1** to **16**. This is the number of input channels activated for conversion. The first channel of the scan is always and input channel with the number 1 and the last channel has the number **n**.

### **Vector input thermocouple types (cell array of char)**

For each acquired channel, enter a valid type of either 'J', 'K', 'E', 'T', 'R', 'S', or 'B'. This vector defines the type of thermocouple for each channel. The vector must be the same length as the **Number of channels to be acquired**.

### **Vector of input gains (double array)**

For each acquired channel, enter a valid input gain of either 1, 125, 166.7, or 400. This vector defines the input gain for each

# Measurement Computing PCI-DAS-TC Thermocouple

---

channel. The vector must be the same length as the **Number of channels to be acquired**.

## **Vector or temperature formats (cell array of char)**

For each acquired channel, enter a valid format of either 'C' or 'F'. 'C' = Celsius and 'F' = Fahrenheit. The vector must be the same length as the **Number of channels to be acquired**.

## **Read and output CJC temperature**

If you want the block to read, convert, and output the temperature of the cold junction (CJC) sensor on the board, select this check box. If selected, the block shows an additional output port with the value of the CJC temperature.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpci
```

# Measurement Computing PCI-DAS-TC Thermocouple

---

---

**Note** Each time a target application containing this driver block is downloaded to the target PC, the board automatically does a full calibration. Thermocouple sensor calibration is an extensive procedure and because it has to take place for each channel independently, the calibration time can easily exceed several seconds, especially when the number of channels to be acquired is 5 or higher. Because of this long calibration period during the initialization stage of the target application, the download procedure can time-out and return an error message. To avoid this error, increase the default timeout duration. See “Increasing the Time-Out Value”.

---

# MPL

---

This chapter describes the MPL board supported by xPC Target (<http://www.mpl.ch>).

## MPL PATI

An I/O board with eight differential 10-bit analog and digital channels, 32 32-bit TPU channels, 2 serial interfaces, and 2 CAN interfaces.

## **Boards and Blocks**

## Board

MPL PATI

## General Description

The MPL PowerPC controlled Analog and Timing I/O Intelligence (PATI) has 32 32-bit TPU channels, which xPC Target supports for PWM measurement, PWM generation, time base selection, incremental encoder measurement, and digital I/O. It is based on the Motorola® MPC555 processor. xPC Target supports only the TPU channels for this board. xPC Target supports the MPL PATI board with these driver blocks:

xPC Target supports this board with these driver blocks:

- MPL PATI Digital Input
- MPL PATI Digital Output
- MPL PATI Incremental Encoder
- MPL PATI PWM generate
- MPL PATI PWM measure
- MPL PATI Timebase Setup

When installing MPL PATI boards, note the following slot selection configuration settings. In the following, the first number after the S indicates the switch number (1) and the second number indicates the switch.

- S1-1 and S1-2 — Select the stacking position in which the board is found. Use the following settings:
  - First board: S1-1 — off, S1-2 — off
  - Second board: S1-1 — on, S1-2 — off
  - Third board: S1-1 — off, S1-2 — on
  - Fourth board: S1-1 — on, S1-2 — on
- S3 — off
- S4 — on (enable reboot)
- S5 — off

# MPL PATI

---

- S6 — off
- S7 — off
- S8 — off

See the MPL PATI user documentation for a description of the DIP switch S1. For some target PC CPU boards, you might be limited to two PC/104+ boards, for others, you might be able to add up to four PC/104+ boards.

## **Board Characteristics**

Board name	MPL PATI
Manufacturer	MPL
Bus type	PCI/104
Multiple block instance support	Yes
Multiple board support	Yes



**Purpose** MPL PATI Digital Input block

**Library** xPC Target Library for MPL

**Note** The individual bits of the MPL PATI digital input are in different registers. As a result, you cannot read multiple bits at the same time. To emphasize this, you must use a separate MPL PATI Digital Input block to read each bit.

**Block Parameters**

**Channel**

From the list, select from 1 to 32. This is the single TPU channel that this block reads.

**Channel priority**

From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

**Sample time**

Enter the base sample time or a multiple of the base sample time.

**PCI slot**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# MPL PATI Digital Output

---

**Purpose** MPL PATI Digital Output block

**Library** xPC Target Library for MPL

**Note** The individual bits of the MPL PATI digital output are in different registers. As a result, you cannot update multiple digital outputs at the same time. To emphasize this, you must use a separate MPL PATI Digital Output block to control each bit.

## Block Parameters

### Channel

From the list, select from 1 to 32. This is the single TPU channel that this block controls.

### Channel priority

From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

### Reset

This parameter controls the behavior of the bit at model termination. Enter one of the following values:

- 1

Resets the output bit to the value in **Initial Value**.

- 0

Leaves the output bit at the most recent value attained while the model was running.

### Initial value

Enter the initial voltage value for the output bit. The output bit is set to this value between the time the model is downloaded and the time the model is started. Also, if the **Reset** parameter has a value of 1, the output bit is reset to the initial value when the model is stopped.

### Sample time

Enter the base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# MPL PATI Incremental Encoder

---

**Purpose** MPL PATI Incremental Encoder block

**Library** xPC Target Library for MPL

**Note** This block has one input port and one output port.

- Input port

Selects the counting mode. The block M input port determines the counting mode. The incremental encoder function has two counting modes, normal and fast. In fast mode, the count direction is not controlled by the polarity of the two signals but continues the same direction that was determined when the counter was last in normal mode. The maximum counting rate is much higher in fast mode than in normal mode.

- Normal mode

If the block input M is less than 0.5, the counter works in normal mode and counts on each edge. The edge sense and polarity of each signal determines the count direction of each signal. Each rising and falling edge of both inputs is counted. You must set the counting mode to normal when the model is first started. You can change it to fast mode afterward.

- Fast mode

If the block input M is greater than or equal to 0.5, the counter works in fast mode. The counter increments by 4 each time the primary channel has a rising edge.

- Output port

Returns the current position count as a double.

To keep the counting mode set to normal mode, just connect a constant block with the value 0 to the M input.

## Block Parameters

### Channel pair

From the list, choose a pair of sequential channels for the incremental encoder. Both channels must be in the same group of 16. For example, 1+2, 2+3, or 18+19. If you want to use channels 16 or 32, their sequential partners are 1 and 17, for example, 16+1 and 32+17.

### Channel priority

From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

### Initial count

The initial count specifies the initial value for the encoder count register when the model starts executing.

### Output range

From the list, choose one of the following. This block converts the output to doubles.

- signed [-32768 +32767]

Interprets the 16-bit count as a signed quantity. This places the discontinuity far from the 0 origin. Counting up from 32767 goes to -32768 on the next count.

- unsigned [0 65535]

Interprets the 16-bit count as an unsigned quantity. This places the discontinuity at the origin, where counting up from 65535 goes to 0 on the next count.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI slot

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## MPL PATI Incremental Encoder

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** MPL PATI PWM generate block

**Library** xPC Target Library for MPL

<b>Scaling Output to Input</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	0 to 5 volts	Double	Counts

**Note** This block uses the PWM module in the time processor units (TPUs) on the board to generate rectangular wave output. See the MPC555 documentation from Freescale Semiconductor at <http://www.freescale.com> for more information on the PWM module.

This block has three inputs:

- H — Number of counts of the chosen time base that the output should remain high. You can use a count from 0 to 32767. With the use of multiple channels, you might not be able to get your output to 0 width smoothly. If you have a small number of channels, you can try to use the channel **Priority** parameter to more smoothly approach 0 width.
- L — Number of counts of the chosen time base that the output should remain low. Range from 0 to 32767. With the use of multiple channels, you might not be able to smoothly get your output to 0 width. If you have a small number of channels, you can try to use the channel **Priority** parameter to more approach 0 width smoothly.
- A — Specifies whether or not the output is to be armed. If the A input is 0, the block stops the output and the output level is the one chosen in the dialog. If A is 1, the output runs. This port does not appear if you do not select **Arm input**.

**Limitations** If you use multiple blocks for different channels, the relative timing between the output signals is not controlled and is not repeatable on the order of a few 10s of microseconds even if the output frequency is the same.

# MPL PATI PWM generate

---

## Block Parameters

### Channel

From the list, select a channel from 1 to 32. This board has two TPUs. Channels 1 through 16 are on the first TPU, channels 17 through 32 are on the second TPU. The channel you select works with the **Timebase** parameter.

### Priority

From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

### Initial high count

Enter the high count to be used from initialization until the model is running.

### Initial low count

Enter the low count to be used from initialization until the model is running.

### Immediate update

Select this check box to force the output to use the new count immediately. Selecting this check box can result in very short output pulses. It terminates long pulses immediately.

Clear this check box to allow the current output cycle to finish before the update. Clearing this check box generates smoother frequency and phase transitions without transients.

The block does not interrupt long pulses. Note that your selection of this parameter value depends on your application.

### Arm input

Select this check box to indicate that the ARM input is present. If this check box is selected, the channel behaves as follows. Note that there might be a delay, on the order of the output period, between the time that the ARM input becomes 1 and the output port starts changing.

- If the input signal to this port is 1, the channel outputs the signal specified by the H and L ports.



- If the input signal is to this port is 0, the output stops at the level specified by the **Disarm state** drop-down list.

If you do not select this check box, the block behaves as if the ARM input is 1. No A input port appears.

### **Disarm state**

From the drop-down list, select

- Disarm to LOW
- Disarm to HIGH

When the ARM input is 0, this parameter specifies the state of the output.

### **State on model stop**

From the drop-down list, select

- Leave running

Continue to get output pulses at the last H and L port values.

- Stop to LOW

Immediately go to the LOW output state and maintain that level.

- Stop to HIGH

Immediately go to the HIGH output state and maintain that level.

### **Timebase**

From the list, select TCR1 or TCR2. This depends on the MPL PATI Timebase Setup settings. If you do not use the MPL PATI Timebase Setup block, the TCR1 time base defaults to 1.25 MHz.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# MPL PATI PWM generate

---

## PCI slot

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## See Also

MPL PATI Timebase Setup

**Purpose** MPL PATI PWM measure

**Library** xPC Target Library for MPL

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	0 to 5 volts	Double	Counts of current time base

**Note** This block uses the PTA module in the time processor units (TPUs) on the board to measure full cycle period or high or low time in an input waveform. To properly use the TPUs, you must correctly condition the TPU inputs with a voltage between 0 and 5 volts. The inputs also must have a reasonably fast rise or fall time; otherwise, false triggering might occur. The inputs use TTL level transitions. See the MPC555 documentation from Freescale Semiconductor at (<http://www.freescale.com>) for more information on the PTA module. This block outputs the number of counts of the input time base (default 20 MHz, configurable in the MPL PATI Timebase Setup block) that meet the mode chosen.

## Block Parameters

### Channel

From the list, select a channel from 1 to 32. This board has two TPUs. Channels 1 through 16 are on the first TPU, channels 17 through 32 are on the second TPU. The channel you select works with the time base parameter.

### Channel priority

From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

### Mode

From the list, select

- Total High Time — For measuring pulse widths

# MPL PATI PWM measure

---

- Total Low Time — For measuring pulse widths
- Between Rising Edges — For period measurements
- Between Falling Edges — For period measurements

## Number of periods to count

Enter the number of periods to count. To measure every cycle separately, enter a value of 1 (default). To get higher precision by averaging, enter a higher value, then divide the block output by this value. For example, to gain a period that is accurate to 0.1 count, enter a value of 10 for this count, then divide the block output by 10.

## Timebase

From the list, select TCR1 or TCR2. This depends on the MPL PATI Timebase Setup settings. If you do not use the MPL PATI Timebase Setup block, the TCR1 time base defaults to 1.25 MHz.

## Sample time

Enter the base sample time or a multiple of the base sample time.

## PCI slot

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## See Also

MPL PATI Timebase Setup

**Purpose** MPL PATI Timebase Setup block

**Library** xPC Target Library for MPL

**Note** The Motorola MPC555 processor contains two time processor units (TPUs). Each TPU has 16 timer channels (1 to 16 and 17 to 32), totaling 32 channels, for one MPL PATI board. Each TPU has two time bases (four between the two TPUs) that can be used in the timer functions. The basic frequency for the TPU is the clock frequency for the CPU, (40 MHz). The minimum divider that can be applied is a factor of 2, which gives the maximum timer frequency of 20 MHz. This block allows you to set the frequency for each of the four time bases, two time bases for channels 1 to 16 and two for channels 17 to 32.

Each TPU allows two different timer sources called TCR1 and TCR2.

- TCR1 is driven by a clock source with a frequency of  $40 \text{ MHz}/N$ , where  $N$  is a power of 2 between 2 and 512.
- TCR2 can be driven either by an external clock source or by a clock source with a frequency of  $40 \text{ MHz}/N$ , where  $N$  is a power of 2 between 8 and 32. Note that the MPL PATI user documentation refers to the channel's 17 to 32 external input as TPU\_T2CLKB.

The hardware architecture restricts allowable values for the time bases. See the MPC555 documentation from Freescale Semiconductor at <http://www.freescale.com> for more information on the TPUs and time bases.

## Block Parameters

### TCR1 divider (1–16)

From the list, choose the divider that sets the TCR1 time base for channels 1 to 16. The minimum divider of 2 gives the maximum time base frequency of 20 MHz. The maximum divider of 512 gives a time base frequency of 78.125 kHz.

### TCR2 divider (1–16)

From the list, choose one of the following:

# MPL PATI Timebase Setup

---

- 8 (5 Mhz) or ext
- 16 (2.5 Mhz) or ext/2
- 32 (1.25 Mhz) or ext/4
- 64 (625 Khz) or ext/8

This divider sets the TCR2 time base for channels 1 to 16. You can specify that the TCR2 time base can be driven by either the 40 MHz CPU clock or by a signal on the external T2CLK input pin. Note that the maximum frequency of the external signal is approximately  $40 \text{ MHz}/9 = 4.44 \text{ MHz}$ .

If you want to use the external clock, select the **TCR2 external clock input (1-16)** check box.

## **TCR2 external clock input (1-16)**

Select this check box to use the signal on the T2CLK pin as the source for TCR2. Use this parameter with the **TCR2 divider (1-16)** parameter.

## **TCR1 divider (17-32)**

From the list, choose the divider that sets the TCR1 time base for channels 17 to 32. A minimum divider of 2 gives the maximum time base frequency of 20 MHz. A maximum divider of 512 gives a time base frequency of 78.125 kHz.

## **TCR2 divider (17-32)**

From the list, choose one of the following:

- 8 (5 Mhz) or ext
- 16 (2.5 Mhz) or ext/2
- 32 (1.25 Mhz) or ext/4
- 64 (625 Khz) or ext/8

This divider sets the TCR2 time base for channels 1 to 16. You can specify that the TCR2 time base can be driven by either the 40 MHz CPU clock or by a signal on the external T2CLK input

pin. Note that the maximum frequency of the external signal is approximately  $40 \text{ MHz}/9 = 4.44 \text{ MHz}$ .

If you want to use the external clock, select the **TCR2 external clock input (17-32)** check box.

### **TCR2 external clock (17-32)**

Select this check box to use the signal on the T2CLK pin as the source for TCR2. Use this parameter with the **TCR2 divider (17-32)** parameter.

### **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```





# National Instruments

---

This chapter describes National Instruments I/O boards supported by xPC Target (<http://www.ni.com>).

National Instruments AT-AO-6	I/O board with 6 analog output (D/A) channels, and 16 digital I/O lines.
National Instruments AT-AO-10	I/O board with 10 analog output (D/A) channels, and 16 digital I/O lines.
National Instruments PC-DIO-24	I/O board with 24 digital input and output lines.
National Instruments PC-TIO-10	I/O board with 16 digital input and output lines, and 10 counter/timer channels
National Instruments PCI-6023E	I/O board with 16 single or 8 differential analog input (A/D) channels, 8 digital I/O lines, and 2 counter/timers.
National Instruments PCI-6024E	I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, 8 digital input and output lines, and 2 counter/timers.
National Instruments PCI-6025E	I/O board with 16 single or 8 differential analog inputs (A/D) channels, 2 analog output channels, 32 digital input and output lines, and 2 counter/timers.

National Instruments PCI-6030E (formerly PCI-MIO-16XE-10)	I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, 8 digital input and output lines, and 2 counter/timers.
National Instruments PCI-6031E	I/O board with 64 single or 32 differential analog input (A/D) channels, 2 analog output (D/A) channels, 8 digital input and output lines, and 2 counter/timers.
National Instruments PCI-6040E (formerly PCI-MIO-16E-4)	I/O board with 16 single or 8 differential analog input channels, 2 analog output channels, 8 digital input and output lines, and 2 counter/timers.
National Instruments PCI/PXI-6052E	I/O board with 16 single or 8 differential analog input channels, 2 analog output channels, and 8 digital input and output lines.
National Instruments PCI-6070E (formerly PCI-MIO-16E-1)	I/O board with 16 single or 8 differential analog input channels, 2 analog output channels, 8 digital input and output lines, and 2 counter/timers.
National Instruments PCI-6071E	I/O board with 64 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, 8 digital input and output lines, and 2 counter/timers.
National Instruments PCI-6503	I/O board with 24 digital input and output lines.
National Instruments PCI-6527	I/O board with 24 filtered digital input lines and 24 digital output lines. You can individually turn input filtering on or off for each of the 24 input lines.

National Instruments PCI-6601	General purpose counter/timer board. It has four 32-bit counter channels.
National Instruments PCI/PXI-6602	General purpose counter/timer board. It has eight 32-bit counter channels.
National Instruments PCI-6703	I/O board with 16 voltage outputs and 8 digital I/O lines.
National Instruments PCI-6704	I/O board with 16 voltage outputs, 16 current outputs, and 8 digital I/O lines.
National Instruments PCI/PXI-6711	I/O board with 8 analog output (D/A) channels and 8 digital input and output lines.
National Instruments PCI/PXI-6713	I/O board with 4 analog output (D/A) channels and 8 digital input and output lines.
National Instruments PCI/PXI-6731	I/O board with 4 analog output (D/A) channels (16-bit) and 8 digital input and output lines.
National Instruments PCI/PXI-6733	I/O board with 8 analog output (D/A) channels (16-bit) and 8 digital input and output lines.
National Instruments PCI-DIO-96	I/O board with 96 digital input and output lines.
National Instruments PXI-6040E	I/O board with 16 single or 8 differential analog input channels, 2 analog output channels, 8 digital input and output lines, and 2 counter/timers.
National Instruments PXI-6070E	I/O board with 16 single or 8 differential analog input channels, 2 analog output channels, 8 digital input and output lines, and 2 counter/timers.

National Instruments PXI-6071E	I/O board with 64 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, 8 digital input and output lines, and 2 counter/timers.
National Instruments PXI-6508	I/O board with 96 digital input and output lines.
National Instruments PXI-6527	I/O board with 24 filtered digital input lines and 24 digital output lines. You can individually turn input filtering on or off for each of the 24 input lines.
National Instruments PXI-6704	I/O board with 16 voltage outputs, 16 current outputs, and 8 digital I/O lines.
“Boards and Blocks — Alphabetical List” on page 26-5	Description of block parameters for National Instruments driver blocks.

## **Boards and Blocks – Alphabetical List**

# National Instruments AT-AO-6

---

**Board** National Instruments AT-AO-6

**General Description** The AT-AO-6 is an I/O board with 6 analog output (D/A) channels (12-bit), and 16 digital I/O lines.

xPC Target supports this board with this driver block:

- National Instruments AT-AO-6 Analog Output (D/A)

<b>Board Characteristics</b>	Board name	AT-AO-6
	Manufacturer	National Instruments
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# National Instruments AT-AO-6 Analog Output (D/A)

**Purpose** AT-AO-6 Analog Output block

**Library** xPC Target Library for National Instruments

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 6. 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

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

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

[-10,10]

# National Instruments AT-AO-6 Analog Output (D/A)

---

The range settings must correspond to 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.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



**Board** National Instruments AT-AO-10

**General Description** The AT-AO-10 is an I/O board with 10 analog output (D/A) channels (12-bit), and 16 digital I/O lines.

xPC Target supports this board with this driver block:

- National Instruments AT-AO-10 Analog Output (D/A)

<b>Board Characteristics</b>	Board name	AT-AO-10
	Manufacturer	National Instruments
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# National Instruments AT-AO-10 Analog Output (D/A)

**Purpose** AT-AO-10 Analog Output block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 10. 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**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# National Instruments AT-AO-10 Analog Output (D/A)

---

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

[ -10, 10]

The range settings must correspond to 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.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# National Instruments PC-DIO-24

---

**Board** National Instruments PC-DIO-24

**General Description** The PC-DIO-24 is an I/O board with 24 digital input and output lines. xPC Target supports this board with these driver blocks:

- National Instruments PC-DIO-24 Digital Input
- National Instruments PC-DIO-24 Digital Output

<b>Board Characteristics</b>	Board name	PC-DIO-24
	Manufacturer	National Instruments
	Bus type	ISA
	Access method	I/O-mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# National Instruments PC-DIO-24 Digital Input

**Purpose** PC-DIO-24 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** The PC-DIO24 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# National Instruments PC-DIO-24 Digital Input

---

## **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

# National Instruments PC-DIO-24 Digital Output

**Purpose** PC-DIO-24 Digital Output

**Library** xPC Target Library for National Instruments

**Note** The PC-DIO24 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

$< 0.5 = \text{TTL low}$

$\geq 0.5 = \text{TTL high}$

## 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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# National Instruments PC-DIO-24 Digital Output

---

## **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



**Board** National Instruments PC-TIO-10

**General Description** The PC-TIO-10 is an I/O board with 16 digital input and output lines, and 10 counter/timer channels (16-bit).

xPC Target supports this board with these driver blocks:

- National Instruments PC-TIO-10 Digital Input
- National Instruments PC-TIO-10 Digital Output
- National Instruments PC-TIO-10 Counter PWM
- National Instruments PC-TIO-10 Counter PWM & ARM
- National Instruments PC-TIO-10 Counter FM
- National Instruments PC-TIO-10 Counter FM & ARM
- National Instruments PC-TIO-10 PWM Capture
- National Instruments PC-TIO-10 FM Capture
- National Instruments PC-TIO-10xx

<b>Board Characteristics</b>	Board Name	PC-TIO10
	Manufacturer	National Instruments
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# National Instruments PC-TIO-10 Digital Input

---

**Board** PC-TIO-10 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** The PC-TIO-10 has one MC6821 chip with 2 ports (PIA A, PIA B). Each port has a maximum of 8 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, the port is configured as input.

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

## **Block Parameters**

### **Number of channels**

Enter the number of digital input channels (from 1 to 8) to use with the port specified in **Port**.

### **Port**

From the list choose either PIA A, or PCA B. The I/O board has a MC6821 chip with 2 ports. The **Port** parameter defines which port of the MC6821 chip is used for this driver block. Each port has a maximum of 8 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.

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

# National Instruments PC-TIO-10 Digital Input

---

## **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

# National Instruments PC-TIO-10 Digital Output

---

**Purpose** PC-TIO-10 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** The PC-TIO-10 has one MC6821 chip with 2 ports (PIA A, PIA B). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	$< 0.5 = \text{TTL low}$ $\geq 0.5 = \text{TTL high}$

## **Block Parameters**

### **Number of channels**

Enter the number of digital output channels (from 1 to 8) to use with the port specified in **Port**.

### **Port**

From the list choose either PIA A, or PCA B. The I/O board has a MC6821 chip with 2 ports. The **Port** parameter defines which port of the MC6821 chip is used for this driver block. Each port has a maximum of 8 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.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# National Instruments PC-TIO-10 Digital Output

---

## **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

# National Instruments PC-TIO-10 Counter PWM

---

**Purpose** PC-TIO-10 Counter PWM block

**Library** xPC Target Library for National Instruments

**Note** The PC-TIO-10 has two AM9513A chips each with 5 counters for a total of 10 counters on the board.

The PC-TIO-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 which defines the variable duty cycle between 0 and 1. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **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.

### **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 has to be 17.5 kHz, then choose F2=100kHz as the Frequency base and enter 0.175 as the **Relative output frequency**.  $100 \text{ kHz} \times 0.175 = 17.5 \text{ kHz}$

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and gets 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

# National Instruments PC-TIO-10 Counter PWM & ARM

**Purpose** PC-TIO-10 Counter PWM & ARM block

**Library** xPC Target Library for National Instruments

**Note** The PC-TIO-10 has two AM9513A chips each with 5 counters for a total of 10 counters on the board.

The PC-TIO-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 to arm and disarm the counter by the second block input. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	TTL	Duty cycle: Double Arm: Double	0 to 1 <0.5 disarmed ≥0.5 armed

## **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.

### **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 has to be 17.5 kHz, then choose F2=100kHz as the **Frequency base** and enter 0.175 as the **Relative output frequency**.  $100 \text{ kHz} \times 0.175 = 17.5 \text{ kHz}$



## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running. If the application is running, the second input port controls whether the counter is armed or disarmed. This parameter sets the TTL level when the counter is disarmed.

## **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

## **Base address**

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

0x300

# National Instruments PC-TIO-10 Counter FM

---

**Purpose** PC-TIO-10 Counter FM block

**Library** xPC Target Library for National Instruments

**Note** The PC-TIO-10 has two AM9513A chips each with 5 counters for a total of 10 counters on the board.

The PC-TIO-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 PWM signal is output at the pin named OUT.

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **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.

### **Output duty Cycle**

Enter a value between 0 and 1 to set the duty cycle of the square wave. The Duty Cycle is held fixed during execution of the target application.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

### **Level sequence of square wave**

From the list, choose either high-low or low-high:

# National Instruments PC-TIO-10 Counter FM

---

- 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 gets 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

# National Instruments PC-TIO10 Counter FM & ARM

---

**Purpose** PC-TIO10 Counter FM & ARM block

**Library** xPC Target Library for National Instruments

**Note** The PC-TIO-10 has two AM9513A chips each with 5 counters for a total of 10 counters on the board.

The PC-TIO-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 to arm and disarm the counter by the second block input. For the corresponding counter channel, the PWM signal is output at the pin named OUT.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Variable frequency: Double Arm: Double	<0.5 disarmed ≥0.5 armed

## 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.

### Output duty Cycle

Enter a value between 0 and 1 to set the duty cycle of the square wave. The Duty Cycle is held fixed during execution of the target application.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the **Output duty cycle** defined in the setting above define 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. If a value 1 is asserted, the counter gets armed.

## **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 corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# National Instruments PC-TIO10 PWM Capture

---

**Purpose** PC-TIO10 PWM Capture block

**Library** xPC Target Library for National Instruments

**Note** This block programs the AMD9513A for capturing PWM signals by using two counters. One counter measures the cycle duration, and the other counter measures the duration 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 has to enter the pins named GATE of both corresponding counter channels (parallel wiring). Both CLK pins have to be left unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1&2, 2&3, 3&4, 4&5, 6&7, 7&8, 8&9, 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 CIO-CTR05 has to be in position 1 MHz not 5 MHz.

# National Instruments PC-TIO10 PWM Capture

---

## **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

# National Instruments PC-TIO10 FM Capture

---

**Purpose** PC-TIO10 FM Capture block

**Library** xPC Target Library for National Instruments

**Note** This block programs the AMD9513A 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 has to enter the pin named GATE of the corresponding counter channel. The CLK pin has to be left unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, 4, or 5. 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 CIO-CTR05 has to 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)



# National Instruments PC-TIO10 FM Capture

---

## **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

# National Instruments PC-TIO-10xx

---

**Purpose** PC-TIO-10xx block

**Library** xPC Target Library for National Instruments

**Note** You can use this block to program the AMD9513A 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.

# National Instruments PCI-6011E (formerly PCI-MIO-16XE-50)

---

## Board

National Instruments PCI-6011E

## General Description

This board was formerly PCI-MIO-16XE-50.

The PCI-6011E is an I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 100 kHz, 2 analog output (D/A) channels (12-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6011E Analog Input (A/D) (formerly PCI-MIO-16XE-50)
- National Instruments PCI-6011E Analog Output (D/A) (formerly PCI-MIO-16XE-50)
- National Instruments PCI-6011E Digital Input (formerly PCI-MIO-16XE-50)
- National Instruments PCI-6011E Digital Output (formerly PCI-MIO-16XE-50)
- National Instruments PCI-6011E Pulse Generation (formerly PCI-MIO-16XE-50)
- National Instruments PCI-6011E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-50)

# National Instruments PCI-6011E (formerly PCI-MIO-16XE-50)

---

## Board Characteristics

Board name	PCI-6011E
Manufacturer	National Instruments
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes

# National Instruments PCI-6011E Analog Input (A/D) (formerly PCI-MIO-16XE-50)

**Purpose** PCI-6011E Analog Input block

**Library** xPC Target Library for National Instruments

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each channel in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table lists the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-1 to + 1	-1
-0.1 to +0.1	-0.1
0 to +10	10

# National Instruments PCI-6011E Analog Input (A/D) (formerly PCI-MIO-16XE-50)

Input Range (V)	Range Code
0 to +5	5
0 to +1	1
0 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are 0 to +1 volts, enter

[ -10, 1, 1 ]

## Input coupling vector

Enter a coupling code for each channel in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table lists the couplings for this driver and the corresponding coupling codes.

Coupling	Coupling Code	Description
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.

# National Instruments PCI-6011E Analog Input (A/D) (formerly PCI-MIO-16XE-50)

---

Coupling	Coupling Code	Description
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input eight channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

### Sample time

Model base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## National Instruments PCI-6011E Analog Input (A/D) (formerly PCI-MIO-16XE-50)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6011E Analog Output (D/A) (formerly PCI-MIO-16XE-50)

---

**Purpose** PCI-6011E Analog Output block

**Library** xPC Target Library for National Instruments

**Note** The analog output range of this board is set -10 to +10 volts.

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### **Reset vector**

The reset vector controls the behavior for each 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

# National Instruments PCI-6011E Analog Output (D/A) (formerly PCI-MIO-16XE-50)

---

the initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6011E Digital Input (formerly PCI-MIO-16XE-50)

---

**Purpose** PCI-6011E Digital Input block

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## National Instruments PCI-6011E Digital Input (formerly PCI-MIO-16XE-50)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6011E Digital Output (formerly PCI-MIO-16XE-50)

**Purpose** PCI-6011E Digital Output block

**Library** xPC Target Library for National Instruments

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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

# National Instruments PCI-6011E Digital Output (formerly PCI-MIO-16XE-50)

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6011E Pulse Generation (formerly PCI-MIO-16XE-50)

---

**Purpose** PCI-6011E Pulse Generation block

**Library** xPC Target Library for National Instruments

**Requirements** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6011E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-50)

**Purpose** PCI-6011E Pulse Width/Period Measurement

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified measurement. When measuring pulse width, the output reflects



# National Instruments PCI-6011E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-50)

---

the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6023E

---

**Board** National Instruments PCI-6023E

## General Description

The PCI-6023E is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 200 kHz, 8 digital I/O lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6023E Analog Input (A/D)
- National Instruments PCI-6023E Digital Input
- National Instruments PCI-6023E Digital Output
- National Instruments PCI-6023E Pulse Generation
- National Instruments PCI-6023E Pulse Width/Period Measurement

## Board Characteristics

Board name	PCI-6023E
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	A/D: No, Digital I/O: Yes; Period/Pulsewidth Measurement: Yes; Pulsetrain Generation: No.
Multiple board support	Yes

# National Instruments PCI-6023E Analog Input (A/D)

**Purpose** PCI-6023E Analog Input block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-0.5 to +0.5	-0.5
-0.05 to +0.05	-0.05

# National Instruments PCI-6023E Analog Input (A/D)

For example, if the first channel is -10 to + 10 volts and the second and fifth channels are -5 to +5 volts, enter

[-10, -5, -5]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

Coupling	Coupling Code	Description
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

# National Instruments PCI-6023E Analog Input (A/D)

---

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## **Sample time**

Model base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6023E Digital Input

---

**Purpose** PCI-6023E Digital Input

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI-6023E Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6023E Digital Output

---

**Purpose** PCI-6023E Digital Output

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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



# National Instruments PCI-6023E Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6023E Pulse Generation

---

**Purpose** PCI-6023E Pulse Generation

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6023E Pulse Width/Period Measurement

**Purpose** PCI-6023E Pulse Width/Period Measurement

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column is the result of a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified

# National Instruments PCI-6023E Pulse Width/Period Measurement

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board

National Instruments PCI-6024E

## General Description

The PCI-6024E is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 200 kHz, 2 analog output (D/A) channels (12-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6024E Analog Input (A/D)
- National Instruments PCI-6024E Analog Output (D/A)
- National Instruments PCI-6024E Digital Input
- National Instruments PCI-6024E Digital Output
- National Instruments PCI-6024E Pulse Generation
- National Instruments PCI-6024E Pulse Width/Period Measurement

## Board Characteristics

Board name	PCI-6024E
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes

# National Instruments PCI-6024E Analog Input (A/D)

**Purpose** PCI-6024E Analog Input block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-0.5 to +0.5	-0.5
-0.05 to +0.05	-0.05

# National Instruments PCI-6024E Analog Input (A/D)

For example, if the first channel is -10 to + 10 volts and the second and fifth channels are -5 to +5 volts, enter

[-10, -5, -5]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

Coupling	Coupling Code	Description
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

# National Instruments PCI-6024E Analog Input (A/D)

---

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## **Sample time**

Model base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6024E Analog Output (D/A)

**Purpose** PCI-6024E Analog Output block

**Library** xPC Target Library for National Instruments

**Note** The analog output range of this board is set -10 to +10 volts.

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

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### **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

# National Instruments PCI-6024E Analog Output (D/A)

---

the initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6024E Digital Input

**Purpose** PCI-6024E Digital Input block

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI-6024E Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6024E Digital Output

**Purpose** PCI-6024E Digital Output

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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

# National Instruments PCI-6024E Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6024E Pulse Generation

---

**Purpose** PCI-6024E Pulse Generation

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6024E Pulse Width/Period Measurement

**Purpose** PCI-6024E Pulse Width/Period Measurement

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified



# National Instruments PCI-6024E Pulse Width/Period Measurement

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6025E

---

**Board** National Instruments PCI-6025E

**General Description**

The PCI-6025E is an I/O board with 16 single or 8 differential analog inputs (A/D) channels (12-bit) with a maximum sample rate of 200 kHz, 2 analog output channels (12-bit), 32 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

The PCI-6025E provides 8 digital input and output lines, the PCI-6025E 8255 provides an additional 24 digital input and output lines.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6025E Analog Input (A/D)
- National Instruments PCI-6025E Analog Output (D/A)
- National Instruments PCI-6025E and PCI-6025E 8255 Digital Input
- National Instruments PCI-6025E Digital Output
- National Instruments PCI-6025E Pulse Generation
- National Instruments PCI-6025E Pulse Width/Period Measurement

**Board Characteristics**

Board name	PCI-6025E
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes

# National Instruments PCI-6025E Analog Input (A/D)

**Purpose** PCI-6025E Analog Input block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-0.5 to +0.5	-0.5
-0.05 to +0.05	-0.05

# National Instruments PCI-6025E Analog Input (A/D)

For example, if the first channel is -10 to + 10 volts and the second and fifth channels are -5 to +5 volts, enter

[-10, -5, -5]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

Coupling	Coupling Code	Description
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

# National Instruments PCI-6025E Analog Input (A/D)

---

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## **Sample time**

Model base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6025E Analog Output (D/A)

---

**Purpose** PCI-6025E Analog Output block

**Library** xPC Target Library for National Instruments

**Note** The analog output range of this board is set -10 to +10 volts.

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

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### **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

# National Instruments PCI-6025E Analog Output (D/A)

---

the initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6025E and PCI-6025E 8255 Digital Input

---

**Purpose** PCI-6025E and PCI-6025E 8255 Digital Input blocks

**Library** xPC Target Library for National Instruments

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

## **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 of 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**

(PCI-6025E 8255) From the list choose either A, B, or C. The I/O board has an 82C55A chip with 3 ports. The port name defines which port of the 82C55A chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs. The PCI-6025E provides 8 digital input lines, the PCI-6025E 8255 provides an additional 24 digital input lines, distributed across the ports A, B, and C. 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.



## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6025E Digital Output

---

**Purpose** PCI-6025E Digital Output block

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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**

(PCI-6025E 8255) From the list choose either A, B, or C. The I/O board has an 82C55A chip with 3 ports. The port name defines which port of the 82C55A chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs. The PCI-6025E provides 8 digital input lines, the PCI-6025E 8255 provides an additional 24 digital input lines, distributed across the ports A, B, and C. 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6025E Pulse Generation

---

**Purpose** PCI-6025E Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6025E Pulse Width/Period Measurement

**Purpose** PCI-6025E Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

## Block Parameters

**Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified

# National Instruments PCI-6025E Pulse Width/Period Measurement

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6030E (formerly PCI-MIO-16XE-10)

---

## Board

National Instruments PCI-6030E

## General Description

This board was formerly PCI-MIO-16XE-10.

The PCI-6030E is an I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 100 kHz, 2 analog output (D/A) channels (16-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6030E Analog Input (A/D) (formerly PCI-MIO-16XE-10)
- National Instruments PCI-6030E Analog Output (D/A) (formerly PCI-MIO-16XE-10)
- National Instruments PCI-6030E Digital Input (formerly PCI-MIO-16XE-10)
- National Instruments PCI-6030E Digital Output (formerly PCI-MIO-16XE-10)
- National Instruments PCI-6030E Pulse Generation (formerly PCI-MIO-16XE-10)
- National Instruments PCI-6030E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-10)

# National Instruments PCI-6030E (formerly PCI-MIO-16XE-10)

---

## Board Characteristics

Board name	PCI-6030E
Manufacturer	National Instruments
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes



# National Instruments PCI-6030E Analog Input (A/D) (formerly PCI-MIO-16XE-10)

**Purpose** PCI-6030E Analog Input block

**Library** xPC Target Library for National Instruments

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2 to +2	-2
-1 to + 1	-1
-0.5 to +0.5	-0.5

# National Instruments PCI-6030E Analog Input (A/D) (formerly PCI-MIO-16XE-10)

---

Input Range (V)	Range Code
-0.2 to +0.2	-0.2
-0.1 to +0.1	-0.1
0 to +10	10
0 to +5	5
0 to +2	2
0 to +1	1
0 to +0.5	0.5
0 to +0.2	0.2
0 to +0.1	0.1

For example, if the first channel is -10 to + 10 volts and the second and fifth channels are 0 to +1 volts, enter

`[-10, 1, 1]`

## **Input coupling vector**

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

# National Instruments PCI-6030E Analog Input (A/D) (formerly PCI-MIO-16XE-10)

Coupling	Coupling Code	Description
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

### Sample time

Model base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## National Instruments PCI-6030E Analog Input (A/D) (formerly PCI-MIO-16XE-10)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6030E Analog Output (D/A) (formerly PCI-MIO-16XE-10)

**Purpose** PCI-6030E Analog Output block

**Library** xPC Target Library for National Instruments

**Note** The analog output range of this board is set -10 to +10 volts.

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# National Instruments PCI-6030E Analog Output (D/A) (formerly PCI-MIO-16XE-10)

---

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

```
[ -10, 10]
```

## **Reset vector**

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

## **Initial value vector**

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

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6030E Digital Input (formerly PCI-MIO-16XE-10)

**Purpose** PCI-6030E Digital Input block

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI-6030E Digital Input (formerly PCI-MIO-16XE-10)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6030E Digital Output (formerly PCI-MIO-16XE-10)

**Purpose** PCI-6030E Digital Output block

**Library** xPC Target Library for National Instruments

## Scaling of Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

$< 0.5$  = TTL low

$\geq 0.5$  = TTL high

## 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 of 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.

### 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

# National Instruments PCI-6030E Digital Output (formerly PCI-MIO-16XE-10)

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6030E Pulse Generation (formerly PCI-MIO-16XE-10)

---

**Purpose** PCI-6030E Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6030E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-10)

**Purpose** PCI-6030E Pulse Width/Period Measurement

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified

# National Instruments PCI-6030E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-10)

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6031E

---

**Board** National Instruments PCI-6031E

**General Description** The PCI-6031E is an I/O board with 64 single or 32 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 100 kHz, 2 analog output (D/A) channels (16-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6031E Analog Input (A/D)
- National Instruments PCI-6031E Analog Output (D/A)
- National Instruments PCI-6031E Digital Input
- National Instruments PCI-6031E Digital Output
- National Instruments PCI-6031E Pulse Generation
- National Instruments PCI-6031E Pulse Width/Period Measurement

<b>Board Characteristics</b>	Board name	PCI-6031E
	Manufacturer	National Instruments
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
	Multiple board support	Yes

# National Instruments PCI-6031E Analog Input (A/D)

**Purpose** PCI-6031E Analog Input block

**Library** xPC Target Library for National Instruments

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 64. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2 to +2	-2
-1 to + 1	-1
-0.5 to +0.5	-0.5

# National Instruments PCI-6031E Analog Input (A/D)

---

Input Range (V)	Range Code
-0.2 to +0.2	-0.2
-0.1 to +0.1	-0.1
0 to +10	10
0 to +5	5
0 to +2	2
0 to +1	1
0 to +0.5	0.5
0 to +0.2	0.2
0 to +0.1	0.1

For example, if the first channel is -10 to + 10 volts and the second and fifth channels are 0 to +1 volts, enter

`[-10, 1, 1]`

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.



# National Instruments PCI-6031E Analog Input (A/D)

<b>Coupling</b>	<b>Coupling Code</b>	<b>Description</b>
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## **Sample time**

Model base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## National Instruments PCI-6031E Analog Input (A/D)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6031E Analog Output (D/A)

**Purpose** PCI-6031E Analog Output block

**Library** xPC Target Library for National Instruments

**Note** The analog output range of this board is set -10 to +10 volts.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# National Instruments PCI-6031E Analog Output (D/A)

---

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

```
[ -10, 10]
```

## **Reset vector**

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

## **Initial value vector**

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

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6031E Digital Input

**Purpose** PCI-6031E Digital Input block

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI-6031E Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6031E Digital Output

**Purpose** PCI-6031E Digital Output block

**Library** xPC Target Library for National Instruments

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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

# National Instruments PCI-6031E Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6031E Pulse Generation

---

**Purpose** PCI-6031E Pulse Generation

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6031E Pulse Width/Period Measurement

**Purpose** PCI-6031E Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified measurement. When

# National Instruments PCI-6031E Pulse Width/Period Measurement

---

measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6040E (formerly PCI-MIO-16E-4)

---

## Board

National Instruments PCI-6040E

## General Description

This was formerly PCI-MIO-16E-4.

The PCI-6040E is an I/O board with 16 single or 8 differential analog input channels (12-bit) with a maximum sample rate of 500 kHz, 2 analog output channels (12-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6040E Analog Input (A/D) (formerly PCI-MIO-16E-4)
- National Instruments PCI-6040E Analog Output (D/A) (formerly PCI-MIO-16E-4)
- National Instruments PCI-6040E Digital Input (formerly PCI-MIO-16E-4)
- National Instruments PCI-6040E Digital Output (formerly PCI-MIO-16E-4)
- National Instruments PCI-6040E Pulse Generation (formerly PCI-MIO-16E-4)
- National Instruments PCI-6040E Pulse Width/Period Measurement (formerly PCI-MIO-16E-4)

## Board Characteristics

Board name	PCI-6040E
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped

# National Instruments PCI-6040E (formerly PCI-MIO-16E-4)

---

Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes

# National Instruments PCI-6040E Analog Input (A/D) (formerly PCI-MIO-16E-4)

---

**Purpose** PCI-6040E Analog Input block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2 to +2	-2
-1 to + 1	-1
-0.5 to +0.5	-0.5

# National Instruments PCI-6040E Analog Input (A/D) (formerly PCI-MIO-16E-4)

Input Range (V)	Range Code
-0.2 to +0.2	-0.2
-0.1 to +0.1	-0.1
-0.05 to +0.05	-0.05
0 to +10	10
0 to +5	5
0 to +2	2
0 to +1	1
0 to +0.5	0.5
0 to +0.2	0.2
0 to +0.1	0.1

For example, if the first channel is -10 to + 10 volts and the second and fifth channels are 0 to +1 volts, enter

[ -10, 1, 1 ]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

# National Instruments PCI-6040E Analog Input (A/D) (formerly PCI-MIO-16E-4)

---

<b>Coupling</b>	<b>Coupling Code</b>	<b>Description</b>
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

### **Sample time**

Model base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.



# National Instruments PCI-6040E Analog Input (A/D) (formerly PCI-MIO-16E-4)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6040E Analog Output (D/A) (formerly PCI-MIO-16E-4)

---

**Purpose** PCI-6040E Analog Output block

**Library** xPC Target Library for National Instruments

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

## Block Parameters

### Channel vector

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

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

[-10,10]

# National Instruments PCI-6040E Analog Output (D/A) (formerly PCI-MIO-16E-4)

---

## **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**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6040E Digital Input (formerly PCI-MIO-16E-4)

---

**Purpose** PCI-6040E Digital Input block

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI-6040E Digital Input (formerly PCI-MIO-16E-4)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6040E Digital Output (formerly PCI-MIO-16E-4)

---

**Purpose** PCI-6040E Digital Output block

**Library** xPC Target Library for National Instruments

## Scaling of Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

< 0.5 =  
TTL low  
≥ 0.5 = TTL  
high

## 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 of 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.

### 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

# National Instruments PCI-6040E Digital Output (formerly PCI-MIO-16E-4)

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6040E Pulse Generation (formerly PCI-MIO-16E-4)

---

**Purpose** PCI-6040E Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6040E Pulse Width/Period Measurement (formerly PCI-MIO-16E-4)

**Purpose** PCI-6040E Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

## Block Parameters

**Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified

# National Instruments PCI-6040E Pulse Width/Period Measurement (formerly PCI-MIO-16E-4)

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board

National Instruments PCI/PXI-6052E

## General Description

The PXI/PCI-6052E is an I/O board with 16 single or 8 differential analog input channels (16-bit) with a maximum sample rate of 333 kHz, 2 analog output channels (16-bit) and 8 digital input and output lines, and two counters/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PXI/PCI-6052E Analog Input (A/D)
- National Instruments PXI/PCI-6052E Analog Output (D/A)
- National Instruments PXI/PCI-6052E Digital Input
- National Instruments PXI/PCI-6052E Digital Output
- National Instruments PXI/PCI-6052E Pulse Generation
- National Instruments PXI/PCI-6052E Pulse Width/Period Measurement

## Board Characteristics

Board name	PXI/PCI-6052E
Manufacturer	National Instruments
Bus type	PCI/PXI
Access method	Memory mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes

# National Instruments PXI/PCI-6052E Analog Input (A/D)

**Purpose** PXI/PCI-6052E Analog Input block

**Library** xPC Target Library for National Instruments

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5

# National Instruments PXI/PCI-6052E Analog Input (A/D)

Input Range (V)	Range Code
-0.2 to +0.2	-0.2
-0.1 to +0.1	-0.1
-0.05 to +0.05	-0.05
0 to +10	10
0 to +5	5
0 to +2	2
0 to +1	1
0 to +0.5	0.5
0 to +0.2	0.2
0 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are 0 to +1 volts, enter

`[-10,1,1]`

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

# National Instruments PXI/PCI-6052E Analog Input (A/D)

Coupling	Coupling Code	Description
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## Sample time

Model base sample time or a multiple of the base sample time.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PXI/PCI-6052E Analog Input (A/D)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI/PCI-6052E Analog Output (D/A)

---

**Purpose** PXI/PCI-6052E Analog Output block

**Library** xPC Target Library for National Instruments

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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



# National Instruments PXI/PCI-6052E Analog Output (D/A)

---

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

[ -10, 10]

## **Reset vector**

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

## **Initial value vector**

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

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI/PCI-6052E Digital Input

---

**Purpose** PXI/PCI-6052E Digital Input block

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PXI/PCI-6052E Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI/PCI-6052E Digital Output

**Purpose** PXI/PCI-6052E Digital Output block

**Library** xPC Target Library for National Instruments

## Scaling of Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

<0.5 =  
TTL low  
≥0.5 = TTL  
high

## 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 of 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.

### 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

# National Instruments PXI/PCI-6052E Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI/PCI-6052E Pulse Generation

---

**Purpose** PXI/PCI-6052E Pulse Generation

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI/PCI-6052E Pulse Width/Period Measurement

**Purpose** PXI/PCI-6052E Pulse Width/Period Measurement

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified

# National Instruments PXI/PCI-6052E Pulse Width/Period Measurement

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6070E (formerly PCI-MIO-16E-1)

---

## Board

National Instruments PCI-6070E

## General Description

This was formerly PCI-MIO-16E-1.

The PCI-6070E is an I/O board with 16 single or 8 differential analog input channels (12-bit) with a maximum sample rate of 1.25 MHz, 2 analog output channels (12-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6070E Analog Input (A/D) (formerly PCI-MIO-16E-1)
- National Instruments PCI-6070E Analog Output (D/A) (formerly PCI-MIO-16E-1)
- National Instruments PCI-6070E Digital Input (formerly PCI-MIO-16E-1)
- National Instruments PCI-6070E Digital Output (formerly PCI-MIO-16E-1)
- National Instruments PCI-6070E Pulse Generation (formerly PCI-MIO-16E-1)
- National Instruments PCI-6070E Pulse Width/Period Measurement (formerly PCI-MIO-16E-1)

## Board Characteristics

Board name	PCI-6070E
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped

# National Instruments PCI-6070E (formerly PCI-MIO-16E-1)

---

Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes

# National Instruments PCI-6070E Analog Input (A/D) (formerly PCI-MIO-16E-1)

**Purpose** PCI-6070E Analog Input block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5

# National Instruments PCI-6070E Analog Input (A/D) (formerly PCI-MIO-16E-1)

---

Input Range (V)	Range Code
-0.2 to +0.2	-0.2
-0.1 to +0.1	-0.1
-0.05 to +0.05	-0.05
0 to +10	10
0 to +5	5
0 to +2	2
0 to +1	1
0 to +0.5	0.5
0 to +0.2	0.2
0 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are 0 to +1 volts, enter

[ -10, 1, 1 ]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

# National Instruments PCI-6070E Analog Input (A/D) (formerly PCI-MIO-16E-1)

Coupling	Coupling Code	Description
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

### Sample time

Model base sample time or a multiple of the base sample time.

# National Instruments PCI-6070E Analog Input (A/D) (formerly PCI-MIO-16E-1)

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6070E Analog Output (D/A) (formerly PCI-MIO-16E-1)

**Purpose** PCI-6070E Analog Output block

**Library** xPC Target Library for National Instruments

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

## Block Parameters

### Channel vector

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

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

[-10,10]

# National Instruments PCI-6070E Analog Output (D/A) (formerly PCI-MIO-16E-1)

---

## **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**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6070E Digital Input (formerly PCI-MIO-16E-1)

---

**Purpose** PCI-6070E Digital Input

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## National Instruments PCI-6070E Digital Input (formerly PCI-MIO-16E-1)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6070E Digital Output (formerly PCI-MIO-16E-1)

**Purpose** PCI-6070E Digital Output

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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

# National Instruments PCI-6070E Digital Output (formerly PCI-MIO-16E-1)

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6070E Pulse Generation (formerly PCI-MIO-16E-1)

---

**Purpose** PCI-6070E Pulse Generation

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6070E Pulse Width/Period Measurement (formerly PCI-MIO-16E-1)

**Purpose** PCI-6070E Pulse Width/Period Measurement

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified

# National Instruments PCI-6070E Pulse Width/Period Measurement (formerly PCI-MIO-16E-1)

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6071E

---

**Board** National Instruments PCI-6071E

**General Description** The PCI-6071E is an I/O board with 64 single or 32 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 1.25 MHz, 2 analog output (D/A) channels (12-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6071E Analog Input (A/D)
- National Instruments PCI-6071E Analog Output (D/A)
- National Instruments PCI-6071E Digital Input
- National Instruments PCI-6071E Digital Output
- National Instruments PCI-6071E Pulse Generation
- National Instruments PCI-6071E Pulse Width/Period Measurement

## Board Characteristics

Board Name	PCI-6071E
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes



# National Instruments PCI-6071E Analog Input (A/D)

**Purpose** PCI-6071E Analog Input (A/D)

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 64. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5

# National Instruments PCI-6071E Analog Input (A/D)

Input Range (V)	Range Code
-0.2 to +0.2	-0.2
-0.1 to +0.1	-0.1
-0.05 to +0.05	-0.05
0 to +10	10
0 to +5	5
0 to +2	2
0 to +1	1
0 to +0.5	0.5
0 to +0.2	0.2
0 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are 0 to +1 volts, enter

[ -10, 1, 1 ]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

# National Instruments PCI-6071E Analog Input (A/D)

Coupling	Coupling Code	Description
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## Sample time

Model base sample time or a multiple of the base sample time.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## National Instruments PCI-6071E Analog Input (A/D)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6071E Analog Output (D/A)

**Purpose** PCI-6071E Analog Output block

**Library** xPC Target Library for National Instruments

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

## Block Parameters

### Channel vector

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

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

[-10,10]

# National Instruments PCI-6071E Analog Output (D/A)

---

## **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**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6071E Digital Input

**Purpose** PCI-6071E Digital Input block

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI-6071E Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6071E Digital Output

**Purpose** PCI-6071E Digital Output block

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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

# National Instruments PCI-6071E Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6071E Pulse Generation

---

**Purpose** PCI-6071E Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6071E Pulse Width/Period Measurement

**Purpose** PCI-6071E Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified

# National Instruments PCI-6071E Pulse Width/Period Measurement

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6503

---

**Board** National Instruments PCI-6503

**General Description** The PCI-6503 is an I/O board with 24 digital input and output lines. xPC Target supports this board with these driver blocks:

- National Instruments PCI-6503 Digital Input
- National Instruments PCI-6503 Digital Output

<b>Board Characteristics</b>	Board name	PCI-6503
	Manufacturer	National Instruments
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# National Instruments PCI-6503 Digital Input

**Purpose** PCI-6503 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** The PCI-6503 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling of Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# National Instruments PCI-6503 Digital Input

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6503 Digital Output

**Purpose** PCI-6503 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** The PCI-6503 has one 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs or outputs

# National Instruments PCI-6503 Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** National Instruments PCI-6527

**General Description** The PCI-6527 is an I/O board with 24 filtered digital input lines and 24 digital output lines. You can individually turn input filtering on or off for each of the 24 input lines. However, there is only one filter time of all of the input lines that have filtering enabled.

xPC Target supports this board with these driver blocks:

- National Instruments PCI-6527 Digital Input
- National Instruments PCI-6527 Digital Output

**Board Characteristics**

Board name	PCI-6527
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI-6527 Digital Input

---

**Purpose** PCI-6527 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** The PCI-6527 has one chip with 3 ports (A,B,C) for digital input. Each port has a maximum of 8 digital I/O lines. Use a separate driver block for each port.

---

**Note** The interrupt on change capability of this board is not used.

---

<b>Scaling of Input to Output</b>	<b>Hardware Input</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	OPTO	Double	$< 0.5 = 0$ $\geq 0.5 = 1$

## **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 of the digital inputs for one port, enter

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

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

### **Port**

From the list choose either A, B, or C. The **Port** parameter defines which port is used for this driver block. Each port has a maximum of 8 digital inputs. In each case, one block is needed for each port.

# National Instruments PCI-6527 Digital Input

---

The documentation for this board from National Instruments labels these ports 0, 1, and 2.

## **Filter vector**

This is a Boolean vector that selects which input lines to filter. For example, if you enter a **Channel vector** of [1, 3, 5], and you want to filter all three lines, enter

[1, 1, 1]

If you want to filter lines 1 and 5, but not line 3, then enter

[1, 0, 1]

If the filter vector is a single element, then it is scalar expanded to the same width as the **Channel vector**. In the example above, if the **Filter vector** is [0], it is expanded to [0, 0, 0]. Likewise, if the **Filter vector** is [1] it is expanded to [1, 1, 1].

## **Filter interval**

Enter the time interval the hardware filter uses to determine a stable pulse. If the input pulse is shorter than this interval, it is ignored. There is only one filter interval for all filtered inputs, and if you filter on more than one digital input block for this board, you must enter the same **Filter interval** for all blocks.

A reasonable value for the **Filter interval** would be 200 to 300 us.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# National Instruments PCI-6527 Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6527 Digital Output

**Purpose** PCI-6527 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** The PCI-6527 has one chip with 3 ports (A,B,C) for digital output. Each port has a maximum of 8 digital output lines. Use a separate driver block for each port.

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	OPTO	Double	<0.5 = OPTO is OFF, no current flowing ≥0.5 = OPTO is ON, current can flow

## **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 of the digital outputs for one port, enter

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

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

### **Port**

From the list choose either A, B, or C. The **Port** parameter defines which port is used for this driver block. Each port has a maximum of 8 digital output lines. In each case, one block is needed for each port.

# National Instruments PCI-6527 Digital Output

---

The documentation for this board from National Instruments labels these ports 3, 4, and 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 a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



**Board** National Instruments PCI-6601

**General Description** The National Instruments PCI-6601 is a general purpose counter/timer board. It has four 32-bit counter channels. The reference frequency for this board is 20 MHz.

xPC Target supports this board with the following driver blocks:

- National Instruments PCI-6601 Digital Input
- National Instruments PCI-6601 Digital Output
- National Instruments PCI-6601 Incremental Encoder
- National Instruments PCI-6601 Pulse Generation
- National Instruments PCI-6601 Pulse Width/Period Measurement
- National Instruments PCI-6601 Armed Pulse Generation

xPC Target does not support the interrupt or timer functionality of the board.

**Board Characteristics**

Board name	PCI-6601
Manufacturer	National Instruments
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# National Instruments PCI-6601 Digital Input

---

**Purpose** PCI-6601 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Only one input block can exist in a model for any given board.

**Block Parameters** **Channel**  
Enter numbers between 1 and 8 to select the digital input ports. This driver allows the selection of individual digital input ports in any order. The number of elements defines the number of digital lines used.

For example, to use the first four ports as digital input channels, enter

[1,2,3,4]

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

---

**Note** Treat the eight I/O ports as two groups of four ports. The two groups are ports 1 to 4 and ports 5 to 8. Do not specify both input and output channels in the same group of four ports. The driver will emit an error.

---

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

**PCI slot**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI-6601 Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6601 Digital Output

---

**Purpose** PCI-6601 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Only one output block can exist in a model for any given board.

**Block Parameters** **Channel**  
Enter numbers between 1 and 8 to select the digital output ports. This driver allows the selection of individual digital output ports in any order. The number of elements defines the number of digital lines used.

For example, to use the last four ports as digital output channels, enter

[5,6,7,8]

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

---

**Note** Treat the eight I/O ports as two groups of four ports. The two groups are ports 1 to 4 and ports 5 to 8. Do not specify both input and output channels in the same group of four ports. The driver will emit an error.

---

## **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, and the **Channel** parameter is longer, 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.

For example, if the block parameters are set as follows:

- **Channel** — [5 6 7 8]
- **Reset vector** — [1 1 0 0]
- **Initial value vector** — [ 1 0 1 0]

Channels 5 and 6 will reset to the initial value vector. Channels 7 and 8 will hold 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.

When execution stops, the driver writes the vector values to the output channels if the corresponding entry in the **Reset vector** is set to 1.

For example, if the block parameters are set as follows:

- **Channel** — [5 6 7 8]
- **Reset vector** — [1 1 0 0]
- **Initial value vector** — [ 1 0 1 0]

When the target application is downloaded to the target, channel 5 is set to 1, channel 6 is set to 0, channel 7 is set to 1 and channel 8 is set to 0.

After the target application executes, then stops, channel 5 will be set back to 1 and channel 6 will be set to 0. Channels 7 and 8 will hold the last values attained.

# National Instruments PCI-6601 Digital Output

---

**Sample time**

The usual sample time in seconds. Can be a MATLAB variable.

**PCI slot**

The usual slot number as used by all PCI board drivers.

# National Instruments PCI-6601 Incremental Encoder

---

**Purpose** PCI-6601 Incremental Encoder block

**Library** xPC Target Library for National Instruments

**Block** **Channel**

**Parameters**

From the list, select a channel number between 1 and 4.

---

**Note** If you want to attach a 3-wire encoder, do so by connecting the channel A signal to the appropriate SOURCE pin, the channel B signal to the AUX pin, and the index signal to the GATE pin.

---

## Counting mode

From the list, select a counting mode. Choose one of the following:

- Normal
- Quadrature Mode X1
- Quadrature Mode X2
- Quadrature Mode X4
- Two-pulse mode
- Synchronous Source Mode

## Initial count

The initial count specifies the initial value for the counter. Enter a nonnegative integer.

## Reload at index pulse

Select this check box to have the count value reset to the value of **Initial count** at each index pulse.

## Index phase

If the **Reload at index pulse** check box is selected, the **Index phase** parameter specifies the phase of the quadrature signals during which the count will be reloaded with the initial count.

# National Instruments PCI-6601 Incremental Encoder

---

The count is reloaded in response to a channel index pulse. From the list, select one of the following:

- A low B low
- A low B high
- A high B low
- A high B high

## **Filter**

You can apply a digital debouncing filter to the input pins prior to processing. From the list, select one of the following filter types:

- None
- Synchronize input to Timebase 3 (20 MHz)
- Minimum pulse width 5 microsec
- Minimum pulse width 1 microsec
- Minimum pulse width 500 nanosec
- Minimum pulse width 100 nanosec
- Minimum pulse width 25 nanosec

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in your target PC, enter

-1

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI-6601 Pulse Generation

---

**Purpose** PCI-6601 Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block are separate, one for the high count and one for the low count.

**Block Parameters**

**Channel**

From the list, choose one of the following:

- 1
- 2
- 3
- 4

These parameters specify the counter(s) to be used with this driver block. In each case, one block is needed for each port.

The reference frequency for this board is 20 MHz.

**Initial high count**

Enter the number of clock ticks the counter should maintain for a high level.

**Initial low count**

Enter the number of clock ticks the counter should maintain at a low level.

**Sample time**

Base sample time or a multiple of the base sample time.

**PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI-6601 Pulse Generation

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6601 Pulse Width/Period Measurement

---

**Purpose** PCI-6601 Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

## Block Parameters

### Channel

From the list, choose one of the following:

- 1
- 2
- 3
- 4

These parameters specify the counter(s) to be used with this driver block. In each case, one block is needed for each port.

### Trigger mode

From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

### Polarity

From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

# National Instruments PCI-6601 Pulse Width/Period Measurement

---

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## Sample time

Base sample time or a multiple of the base sample time.

## PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6601 Armed Pulse Generation

---

**Purpose** PCI-6601 Armed Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block are separate, one for the high count (**H**), one for the low count (**L**), and one for the arm input (**A**). If the arm input signal is less than 0.5, the frequency generation is disabled. If the arm input signal is greater than or equal to 0.5, the frequency generation is enabled.

## Block Parameters

### Channel

From the list, choose one of the following:

- 1
- 2
- 3
- 4

These parameters specify the counter to be used with this driver block. In each case, one block is needed for each port.

The reference frequency for this board is 20 MHz.

### Initial high count

Enter the number of clock ticks the counter should maintain for a high level.

### Initial low count

Enter the number of clock ticks the counter should maintain at a low level.

### Sample time

Base sample time or a multiple of the base sample time.

# National Instruments PCI-6601 Armed Pulse Generation

---

## PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board

National Instruments PCI/PXI-6602

## General Description

The National Instruments PCI/PXI-6602 is a general purpose counter/timer board. It has eight 32-bit counter channels. The reference frequency for this board is 80 MHz.

xPC Target supports this board with the following driver block:

- National Instruments PCI/PXI-6602 Digital Input
- National Instruments PCI/PXI-6602 Digital Output
- National Instruments PCI/PXI-6602 Incremental Encoder
- National Instruments PCI/PXI-6602 Pulse Generation
- National Instruments PCI/PXI-6602 Pulse Width/Period Measurement
- National Instruments PCI/PXI-6602 Armed Pulse Generation

xPC Target does not support the interrupt or timer functionality of the board.

## Board Characteristics

Board name	PCI/PXI-6602
Manufacturer	National Instruments
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# National Instruments PCI/PXI-6602 Digital Input

---

**Purpose** PCI/PXI-6602 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Only one input block can exist in a model for any given board.

**Block Parameters** **Channel**  
Enter numbers between 1 and 8 to select the digital input ports. This driver allows the selection of individual digital input ports in any order. The number of elements defines the number of digital lines used.

For example, to use the first four ports as digital input channels, enter

[1,2,3,4]

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

---

**Note** Treat the eight I/O ports as two groups of four ports. The two groups are ports 1 to 4 and ports 5 to 8. Do not specify both input and output channels in the same group of four ports. The driver will emit an error.

---

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

**PCI slot**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.



# National Instruments PCI/PXI-6602 Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6602 Digital Output

---

**Purpose** PCI/PXI-6602 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Only one output block can exist in a model for any given board.

## **Block Parameters**

### **Channel**

Enter numbers between 1 and 8 to select the digital output ports. This driver allows the selection of individual digital output ports in any order. The number of elements defines the number of digital lines used.

For example, to use the last four ports as digital output channels, enter

[5,6,7,8]

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

---

**Note** Treat the eight I/O ports as two groups of four ports. The two groups are ports 1 to 4 and ports 5 to 8. Do not specify both input and output channels in the same group of four ports. The driver will emit an error.

---

### **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, and the **Channel** parameter is longer, 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.

For example, if the block parameters are set as follows:

- **Channel** — [5 6 7 8]
- **Reset vector** — [1 1 0 0]
- **Initial value vector** — [ 1 0 1 0]

Channels 5 and 6 will reset to the initial value vector. Channels 7 and 8 will hold 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.

When execution stops, the driver writes the vector values to the output channels if the corresponding entry in the **Reset vector** is set to 1.

For example, if the block parameters are set as follows:

- **Channel** — [5 6 7 8]
- **Reset vector** — [1 1 0 0]
- **Initial value vector** — [ 1 0 1 0]

When the target application is downloaded to the target, channel 5 is set to 1, channel 6 is set to 0, channel 7 is set to 1 and channel 8 is set to 0.

After the target application executes, then stops, channel 5 will be set back to 1 and channel 6 will be set to 0. Channels 7 and 8 will hold the last values attained.

# National Instruments PCI/PXI-6602 Digital Output

---

**Sample time**

The usual sample time in seconds. Can be a MATLAB variable.

**PCI slot**

The usual slot number as used by all PCI board drivers.

# National Instruments PCI/PXI-6602 Incremental Encoder

---

**Purpose** PCI/PXI-6602 Incremental Encoder block

**Library** xPC Target Library for National Instruments

**Block Parameters** **Channel**  
From the list, select a channel number between 1 and 8.

---

**Note** If you want to attach a 3-wire encoder, do so by connecting the channel A signal to the appropriate SOURCE pin, the channel B signal to the AUX pin, and the index signal to the GATE pin.

---

## Counting mode

From the list, select a counting mode. Choose one of the following:

- Normal
- Quadrature Mode X1
- Quadrature Mode X2
- Quadrature Mode X4
- Two-pulse mode
- Synchronous Source Mode

## Initial count

The initial count specifies the initial value for the counter. Enter a nonnegative integer.

## Reload at index pulse

Select this check box to have the count value reset to the value of **Initial count** at each index pulse.

## Index phase

If the **Reload at index pulse** check box is selected, the **Index phase** parameter specifies the phase of the quadrature signals during which the count will be reloaded with the initial count.

# National Instruments PCI/PXI-6602 Incremental Encoder

---

The count is reloaded in response to a channel index pulse. From the list, select one of the following:

- A low B low
- A low B high
- A high B low
- A high B high

## **Filter**

You can apply a digital debouncing filter to the input pins prior to processing. From the list, select one of the following filter types:

- None
- Synchronize input to Timebase 3 (80 MHz)
- Minimum pulse width 5 microsec
- Minimum pulse width 1 microsec
- Minimum pulse width 500 nanosec
- Minimum pulse width 100 nanosec
- Minimum pulse width 25 nanosec

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in your target PC, enter

-1

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6602 Pulse Generation

---

**Purpose** PCI/PXI-6602 Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block are separate, one for the high count and one for the low count.

**Block  
Parameters**

**Channel**

From the list, choose one of the following:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

These parameters specify the counter(s) to be used with this driver block. In each case, one block is needed for each port.

The reference frequency for this board is 80 MHz.

**Initial high count**

Enter the number of clock ticks the counter should maintain for a high level.

**Initial low count**

Enter the number of clock ticks the counter should maintain at a low level.

**Sample time**

Base sample time or a multiple of the base sample time.

# National Instruments PCI/PXI-6602 Pulse Generation

---

## PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI/PXI-6602 Pulse Width/Period Measurement

---

**Purpose** PCI/PXI-6602 Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

## Block Parameters

### Counter

From the list, choose one of the following:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

These parameters specify the counter(s) to be used with this driver block. In each case, one block is needed for each port.

### Trigger mode

From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

### Polarity

From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect

# National Instruments PCI/PXI-6602 Pulse Width/Period Measurement

---

one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	20000
Pulse width (high pulse)	Level triggered	Active high	60000
Period	Edge triggered	NA	80000

In every case, the output of the block is the number of clock ticks (of the 80 MHz source clock) required for the specified measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## Filter

From the list, choose one of the following:

- None
- Synchronize input to Timebase 3 (80 MHz)
- Minimum pulse width 5 microsec
- Minimum pulse width 1 microsec
- Minimum pulse width 500 nanosec
- Minimum pulse width 100 nanosec
- Minimum pulse width 25 nanosec

This parameter specifies the level of digital filtering you want to apply to the pin. See the National Instruments PCI/PXI-6602 user manual documentation for details.

# National Instruments PCI/PXI-6602 Pulse Width/Period Measurement

---

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6602 Armed Pulse Generation

---

**Purpose** PCI/PXI-6602 Armed Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block are separate, one for the high count (**H**), one for the low count (**L**), and one for the arm input (**A**). If the arm input signal is less than 0.5, the frequency generation is disabled. If the arm input signal is greater than or equal to 0.5, the frequency generation is enabled.

## Block Parameters

### Channel

From the list, choose one of the following:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

These parameters specify the counter to be used with this driver block. In each case, one block is needed for each port.

The reference frequency for this board is 80 MHz.

### Initial high count

Enter the number of clock ticks the counter should maintain for a high level.

### Initial low count

Enter the number of clock ticks the counter should maintain at a low level.

# National Instruments PCI/PXI-6602 Armed Pulse Generation

---

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6703

---

**Board** National Instruments PCI-6703

**General Description** The PCI-6703 is an I/O board with 16 voltage outputs and 8 digital I/O lines.

xPC Target supports this board with this driver block:

- National Instruments PCI-6703 Analog Output (D/A)

Note, you will notice a variable delay in the voltage output from the board after your target application writes the register on the board. This is because the PCI-6703 has a single D/A converter that is time-share multiplexed through all the outputs. It can take up to 0.9 milliseconds to scan through all channels.

<b>Board Characteristics</b>	Board name	PCI-6703
	Manufacturer	National Instruments
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	No
	Multiple board support	Yes

# National Instruments PCI-6703 Analog Output (D/A)

**Purpose** PCI-6703 Analog Output block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter 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.

### **Sample time**

Base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6704

---

**Board** National Instruments PCI-6704

**General Description**

The PCI-6704 is an I/O board with 16 voltage outputs, 16 current outputs, and 8 digital I/O lines.

xPC Target supports this board with this driver block:

- National Instruments PCI-6704 Analog Output (D/A)

Note, you will notice a variable delay in the voltage output from the board after your target application writes the register on the board. This is because the PCI-6704 has a single D/A converter that is time-share multiplexed through all the outputs. It can take up to 1.8 milliseconds to scan through all channels.

**Board Characteristics**

Board name	PCI-6704
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes



# National Instruments PCI-6704 Analog Output (D/A)

**Purpose** PCI-6704 Analog Output block

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts — Channels 1 to 16	Double	-10 V to 10 V
	Milliamps — Channels 17 to 32	Double	0 ma to 20 ma

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 32. Channels 1-16 refer to the voltage output channels, and 17-32 the current output channels. 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.

### **Sample time**

Base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# National Instruments PCI-6704 Analog Output (D/A)

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board

National Instruments PCI/PXI-6711

## General Description

The PCI/PXI-6711 is an analog output board with four analog output (D/A) channels (12-bit), eight digital input and output lines, and two 24-bit, 20 MHz counters/timers.

xPC Target supports this board with these driver blocks:

- National Instruments PCI/PXI-6711 Analog Output (D/A)
- National Instruments PCI/PXI-6711 Digital Input
- National Instruments PCI/PXI-6711 Digital Output
- National Instruments PCI/PXI-6711 Pulse Generation
- National Instruments PCI/PXI-6711 Pulse Width/Period Measurement

## Board Characteristics

Board name	PCI/PXI-6711
Manufacturer	National Instruments
Bus type	PCI
Multiple block instance support	D/A: No, Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI/PXI-6711 Analog Output (D/A)

**Purpose** PCI/PXI-6711 Analog Output block

**Library** xPC Target Library for National Instruments

**Note** The analog output range of this board is set -10 to +10 volts.

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

## **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 used. For example, to use the analog output channels 1 and 2, enter

[1,2]

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

### **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

# National Instruments PCI/PXI-6711 Analog Output (D/A)

---

the initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6711 Digital Input

---

**Purpose** PCI/PXI-6711 Digital Input block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital input lines used with this block. 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 of the digital inputs with this block, enter

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

Each line can be used for input or output. If a line is not listed in this field, a digital output block using the same board can use that line.

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

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI/PXI-6711 Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6711 Digital Output

---

**Purpose** PCI/PXI-6711 Digital Output

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital output lines used with this block. 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 of the digital outputs with this block, enter

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

Each line can be used for input or output. If a line is not listed in this field, a digital input block using the same board can use that line.

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

### **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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6711 Pulse Generation

---

**Purpose** PCI/PXI-6711 Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6711 Pulse Width/Period Measurement

**Purpose** PCI/PXI-6711 Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Block Parameters** **Counter**  
From the list, select a counter, 0 or 1.

**Trigger mode**

From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**

From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

# National Instruments PCI/PXI-6711 Pulse Width/Period Measurement

---

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board

National Instruments PCI/PXI-6713

## General Description

The PCI/PXI-6713 is an analog output board with eight analog output (D/A) channels (12-bit), eight digital input and output lines, and two 24-bit, 20 MHz counters/timers.

xPC Target supports this board with these driver blocks:

- National Instruments PCI/PXI-6713 Analog Output (D/A)
- National Instruments PCI/PXI-6713 Digital Input
- National Instruments PCI/PXI-6713 Digital Output
- National Instruments PCI/PXI-6713 Pulse Generation
- National Instruments PCI/PXI-6713 Pulse Width/Period Measurement

## Board Characteristics

Board name	PCI/PXI-6713
Manufacturer	National Instruments
Bus type	PCI/PXI
Multiple block instance support	D/A: No, Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI/PXI-6713 Analog Output (D/A)

**Purpose** PCI/PXI-6713 Analog Output block

**Library** xPC Target Library for National Instruments

**Note** The analog output range of this board is set -10 to +10 volts.

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8. 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 analog output channels 1 and 2, enter

[1,2]

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

### **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

# National Instruments PCI/PXI-6713 Analog Output (D/A)

---

the initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6713 Digital Input

---

**Purpose** PCI/PXI-6713 Digital Input

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital input lines used with this block. 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 of the digital inputs with this block, enter

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

Each line can be used for input or output. If a line is not listed in this field, a digital output block using the same board can use that line.

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

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.



# National Instruments PCI/PXI-6713 Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6713 Digital Output

---

**Purpose** PCI/PXI-6713 Digital Output block

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital output lines used with this block. 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 of the digital outputs for with this block, enter

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

Each line can be used for input or output. If a line is not listed in this field, a digital input block using the same board can use that line.

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

### **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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6713 Pulse Generation

---

**Purpose** PCI/PXI-6713 Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6713 Pulse Width/Period Measurement

**Purpose** PCI/PXI-6713 Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Block Parameters** **Counter**  
From the list, select a counter, 0 or 1.

## **Trigger mode**

From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

## **Polarity**

From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

<b>Measurement Objective</b>	<b>Trigger Mode</b>	<b>Polarity</b>	<b>Output</b>
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

# National Instruments PCI/PXI-6713 Pulse Width/Period Measurement

---

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board

National Instruments PCI/PXI-6731

## General Description

The PCI/PXI-6731 is an analog output board with four analog output (D/A) channels (16-bit), eight digital input and output lines, and two 24-bit, 20 MHz counters/timers.

xPC Target supports this board with these driver blocks:

- National Instruments PCI/PXI-6731 Analog Output (D/A)
- National Instruments PCI/PXI-6731 Digital Input
- National Instruments PCI/PXI-6731 Digital Output
- National Instruments PCI/PXI-6731 Pulse Generation
- National Instruments PCI/PXI-6731 Pulse Width/Period Measurement

## Board Characteristics

Board name	PCI/PXI-6731
Manufacturer	National Instruments
Bus type	PCI
Multiple block instance support	D/A: No, Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI/PXI-6731 Analog Output (D/A)

**Purpose** PCI/PXI-6731 Analog Output block

**Library** xPC Target Library for National Instruments

**Note** The analog output range of this board is set -10 to +10 volts.

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

## **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 used. For example, to use the analog output channels 1 and 2, enter

[1,2]

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

### **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



# National Instruments PCI/PXI-6731 Analog Output (D/A)

---

the initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6731 Digital Input

---

**Purpose** PCI/PXI-6731 Digital Input block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital input lines used with this block. 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 of the digital inputs with this block, enter

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

Each line can be used for input or output. If a line is not listed in this field, a digital output block using the same board can use that line.

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

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI/PXI-6731 Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6731 Digital Output

---

**Purpose** PCI/PXI-6731 Digital Output

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital output lines used with this block. 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 of the digital outputs with this block, enter

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

Each line can be used for input or output. If a line is not listed in this field, a digital input block using the same board can use that line.

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

### **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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6731 Pulse Generation

---

**Purpose** PCI/PXI-6731 Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6731 Pulse Width/Period Measurement

**Purpose** PCI/PXI-6731 Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Block Parameters** **Counter**  
From the list, select a counter, 0 or 1.

## **Trigger mode**

From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

## **Polarity**

From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

<b>Measurement Objective</b>	<b>Trigger Mode</b>	<b>Polarity</b>	<b>Output</b>
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

# National Instruments PCI/PXI-6731 Pulse Width/Period Measurement

---

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



## Board

National Instruments PCI/PXI-6733

## General Description

The PCI/PXI-6733 is an analog output board with eight analog output (D/A) channels (16-bit), eight digital input and output lines, and two 24-bit counters.

xPC Target supports this board with these driver blocks:

- National Instruments PCI/PXI-6733 Analog Output (D/A)
- National Instruments PCI/PXI-6733 Digital Input
- National Instruments PCI/PXI-6733 Digital Output
- National Instruments PCI/PXI-6733 Pulse Generation
- National Instruments PCI/PXI-6733 Pulse Width/Period Measurement

## Board Characteristics

Board name	PCI/PXI-6733
Manufacturer	National Instruments
Bus type	PCI
Multiple block instance support	D/A: No, Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI/PXI-6733 Analog Output (D/A)

**Purpose** PCI/PXI-6733 Analog Output block

**Library** xPC Target Library for National Instruments

**Note** The analog output range of this board is set -10 to +10 volts.

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

## **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 used. For example, to use the analog output channels 1 and 2, enter

[1,2]

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

### **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

# National Instruments PCI/PXI-6733 Analog Output (D/A)

---

the initial values between the time the model is downloaded and the time it is started.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6733 Digital Input

---

**Purpose** PCI/PXI-6733 Digital Input block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital input lines used with this block. 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 of the digital inputs with this block, enter

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

Each line can be used for input or output. If a line is not listed in this field, a digital output block using the same board can use that line.

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

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PCI/PXI-6733 Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6733 Digital Output

---

**Purpose** PCI/PXI-6733 Digital Output

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital output lines used with this block. 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 of the digital outputs with this block, enter

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

Each line can be used for input or output. If a line is not listed in this field, a digital input block using the same board can use that line.

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

### **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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI/PXI-6733 Pulse Generation

---

**Purpose** PCI/PXI-6733 Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PCI/PXI-6733 Pulse Width/Period Measurement

**Purpose** PCI/PXI-6733 Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

## **Trigger mode**

From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

## **Polarity**

From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

<b>Measurement Objective</b>	<b>Trigger Mode</b>	<b>Polarity</b>	<b>Output</b>
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

# National Instruments PCI/PXI-6733 Pulse Width/Period Measurement

---

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** National Instruments PCI-DIO-96

**General Description** The PC-DIO-96 is an I/O board with 96 digital input and output lines. xPC Target supports this board with these driver blocks:

- National Instruments PCI-DIO-96 Digital Input
- National Instruments PCI-DIO-96 Digital Output

<b>Board Characteristics</b>	Board name	PC-DIO-96
	Manufacturer	National Instruments
	Bus type	PCI
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# National Instruments PCI-DIO-96 Digital Input

---

**Purpose** PCI-DIO-96 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** The PC-DIO-96 has four 8255 chips with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling of Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has four 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs

# National Instruments PCI-DIO-96 Digital Input

---

depending on which driver block is chosen. In each case, one block is needed for each port.

## **Chip**

From the list choose 1, 2, 3, or 4.

## **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

# National Instruments PCI-DIO-96 Digital Output

---

**Purpose** PCI-DIO-96 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** The PC-DIO24 has four 8255 chips with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs

# National Instruments PCI-DIO-96 Digital Output

---

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.

## **Chip**

From the list choose 1, 2, 3, or 4.

## **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

# National Instruments PXI-6040E

---

## Board

National Instruments PXI-6040E

## General Description

The PXI-6040E is an I/O board with 16 single or 8 differential analog input channels (12-bit) with a maximum sample rate of 500 kHz, 2 analog output channels (12-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PXI-6040E Analog Input (A/D)
- National Instruments PXI-6040E Analog Output (D/A)
- National Instruments PXI-6040E Digital Input
- National Instruments PXI-6040E Digital Output
- National Instruments PXI-6040E Pulse Generation
- National Instruments PXI-6040E Pulse Width/Period Measurement

## Board Characteristics

Board name	PXI-6040E
Manufacturer	National Instruments
Bus type	PXI (Compact PCI)
Access method	Memory mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes



# National Instruments PXI-6040E Analog Input (A/D)

**Purpose** PXI-6040E Analog Input block

**Library** xPC Target Library for National Instruments

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5

# National Instruments PXI-6040E Analog Input (A/D)

Input Range (V)	Range Code
-0.2 to +0.2	-0.2
-0.1 to +0.1	-0.1
-0.05 to +0.05	-0.05
0 to +10	10
0 to +5	5
0 to +2	2
0 to +1	1
0 to +0.5	0.5
0 to +0.2	0.2
0 to +0.1	0.1

For example, if the first channel is -10 to + 10 volts and the second and fifth channels are 0 to +1 volts, enter

`[-10,1,1]`

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

# National Instruments PXI-6040E Analog Input (A/D)

<b>Coupling</b>	<b>Coupling Code</b>	<b>Description</b>
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## **Sample time**

Model base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## National Instruments PXI-6040E Analog Input (A/D)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6040E Analog Output (D/A)

**Purpose** PXI-6040E Analog Output block

**Library** xPC Target Library for National Instruments

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

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

[-10,10]

# National Instruments PXI-6040E Analog Output (D/A)

---

## **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**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6040E Digital Input

**Purpose** PXI-6040E Digital Input block

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PXI-6040E Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PXI-6040E Digital Output

**Purpose** PXI-6040E Digital Output block

**Library** xPC Target Library for National Instruments

## Scaling Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

TTL low = 0.0

TTL high = 1.0

## 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 of 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.

### 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

# National Instruments PXI-6040E Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6040E Pulse Generation

---

**Purpose** PXI-6040E Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6040E Pulse Width/Period Measurement

**Purpose** PXI-6040E Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

## Block Parameters

### Counter

from the list, select a counter, 0 or 1.

### Trigger mode

From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

### Polarity

From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified

# National Instruments PXI-6040E Pulse Width/Period Measurement

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6070E

---

**Board** National Instruments PXI-6070E

**General Description**

The PXI-6070E is an I/O board with 16 single or 8 differential analog input channels (12-bit) with a maximum sample rate of 1.25 MHz, 2 analog output channels (12-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PXI-6070E Analog Input (A/D)
- National Instruments PXI-6070E Analog Output (D/A)
- National Instruments PXI-6070E Digital Input
- National Instruments PXI-6070E Digital Output
- National Instruments PXI-6070E Pulse Generation
- National Instruments PXI-6070E Pulse Width/Period Measurement

**Board Characteristics**

Board name	PXI-6070E
Manufacturer	National Instruments
Bus type	PXI (Compact PCI)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes

# National Instruments PXI-6070E Analog Input (A/D)

**Purpose** PXI-6070E Analog Input block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2 to +2	-2
-1 to + 1	-1
-0.5 to +0.5	-0.5

# National Instruments PXI-6070E Analog Input (A/D)

Input Range (V)	Range Code
-0.2 to +0.2	-0.2
-0.1 to +0.1	-0.1
-0.05 to +0.05	-0.05
0 to +10	10
0 to +5	5
0 to +2	2
0 to +1	1
0 to +0.5	0.5
0 to +0.2	0.2
0 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are 0 to +1 volts, enter

[ -10, 1, 1 ]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.



# National Instruments PXI-6070E Analog Input (A/D)

Coupling	Coupling Code	Description
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## Sample time

Model base sample time or a multiple of the base sample time.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## National Instruments PXI-6070E Analog Input (A/D)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6070E Analog Output (D/A)

**Purpose** PXI-6070E Analog Output block

**Library** xPC Target Library for National Instruments

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

## Block Parameters

### Channel vector

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

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

[-10,10]

# National Instruments PXI-6070E Analog Output (D/A)

---

## **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**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6070E Digital Input

**Purpose** PXI-6070E Digital Input block

**Library** xPC Target Library for National Instruments

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

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# National Instruments PXI-6070E Digital Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6070E Digital Output

**Purpose** PXI-6070E Digital Output block

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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

# National Instruments PXI-6070E Digital Output

---

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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PXI-6070E Pulse Generation

**Purpose** PXI-6070E Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6070E Pulse Width/Period Measurement

**Purpose** PXI-6070E Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

**Block Parameters** **Counter**  
from the list, select a counter, 0 or 1.

**Trigger mode**  
From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

**Polarity**  
From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified

# National Instruments PXI-6070E Pulse Width/Period Measurement

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6071E

---

**Board** National Instruments PXI-6071E

**General Description** The PXI-6071E is an I/O board with 64 single or 32 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 1.25 MHz, 2 analog output (D/A) channels (12-bit), 8 digital input and output lines, and 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz.

xPC Target supports this board with these driver blocks:

- National Instruments PXI-6071E Analog Input (A/D)
- National Instruments PXI-6071E Analog Output (D/A)
- National Instruments PXI-6071E Digital Input
- National Instruments PXI-6071E Digital Output
- National Instruments PXI-6071E Pulse Generation
- National Instruments PXI-6071E Pulse Width/Period Measurement

## Board Characteristics

Board Name	PXI-6071E
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Period/Pulsewidth measurement: Yes; Pulsetrain Generation: No
Multiple board support	Yes

# National Instruments PXI-6071E Analog Input (A/D)

**Purpose** PXI-6071E Analog Input block

**Library** xPC Target Library for National Instruments

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

## Block Parameters

### Channel vector

Enter numbers between 1 and 64. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5

# National Instruments PXI-6071E Analog Input (A/D)

Input Range (V)	Range Code
-0.2 to +0.2	-0.2
-0.1 to +0.1	-0.1
-0.05 to +0.05	-0.05
0 to +10	10
0 to +5	5
0 to +2	2
0 to +1	1
0 to +0.5	0.5
0 to +0.2	0.2
0 to +0.1	0.1

For example, if the first channel is -10 to + +10 volts and the second and fifth channels are 0 to +1 volts, enter

[ -10, 1, 1 ]

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. The coupling vector must be the same length as the channel vector. This driver allows a different coupling for each channel.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

# National Instruments PXI-6071E Analog Input (A/D)

<b>Coupling</b>	<b>Coupling Code</b>	<b>Description</b>
RSE	0	Analog input line connected to the positive input of the PGIA. Analog input ground (AIGND) internally connected to the negative input of the PGIA. See the board manual.
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

[0,0,2]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## **Sample time**

Model base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## National Instruments PXI-6071E Analog Input (A/D)

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# National Instruments PXI-6071E Analog Output (D/A)

**Purpose** PXI-6071E Analog Output block

**Library** xPC Target Library for National Instruments

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

## Block Parameters

### Channel vector

Enter 1 or 2. 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 both of the analog output channels, enter

[1,2]

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

### Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# National Instruments PXI-6071E Analog Output (D/A)

---

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

```
[ -10, 10]
```

## **Reset vector**

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

## **Initial value vector**

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

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6071E Digital Input

**Purpose** PXI-6071E Digital Input block

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	TTL	Double	TTL
			TTL high = 1.0

## **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 of 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# National Instruments PXI-6071E Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6071E Digital Output

**Purpose** PXI-6071E Digital Output block

**Library** xPC Target Library for National Instruments

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **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 of 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.

### **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.

# National Instruments PXI-6071E Digital Output

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6071E Pulse Generation

---

**Purpose** PXI-6071E Pulse Generation block

**Library** xPC Target Library for National Instruments

**Note** The inputs to the block should be a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter should maintain at a low level, and the second element is the number of clock ticks for a high level. For example, to generate a pulse train at 1 kHz with 25% low and 75% high, use the vector

```
[5000 15000]
```

## Block Parameters

### Counter

From the list, choose Counter 0, Counter 1, or Both to select the counter(s) to be used with this driver block. In each case, one block is needed for each physical board.

### Sample time

Base sample time or a multiple of the base sample time.

### PCI slot (-1:autosearch)

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6071E Pulse Width/Period Measurement

**Purpose** PXI-6071E Pulse Width/Period Measurement block

**Library** xPC Target Library for National Instruments

**Note** Connect the signal you want to measure to the GATE input of the chosen counter. The signal must have fast rise and fall times; otherwise, the counter might experience false triggering. Use signal voltage levels between 0 and 5 volts (TTL levels).

## Block Parameters

### Counter

from the list, select a counter, 0 or 1.

### Trigger mode

From the list, choose Level Triggered or Edge Triggered. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column resulted from a 1 kHz pulse train with a 25% low and a 75% high pulse.

### Polarity

From the list, choose Active low or Active high. See the table below for an example of how **Trigger mode** and **Polarity** affect one another. In this table, the data in the Output column is the result of a 1 kHz pulse train with a 25% low and a 75% high pulse.

Measurement Objective	Trigger Mode	Polarity	Output
Pulse width (low pulse)	Level triggered	Active low	5000
Pulse width (high pulse)	Level triggered	Active high	15000
Period	Edge triggered	NA	20000

In every case, the output of the block is the number of clock ticks (of the 20 MHz source clock) required for the specified



# National Instruments PXI-6071E Pulse Width/Period Measurement

---

measurement. When measuring pulse width, the output reflects the number of clock ticks for which the input signal was in the specified (low or high) state. See the table above for an example.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6508

---

**Board** National Instruments PXI-6508

**General Description** The PXI-6508 is an I/O board with 96 digital input and output lines. xPC Target supports this board with these driver blocks:

- National Instruments PXI-6508 Digital Input
- National Instruments PXI-6508 Digital Output

<b>Board Characteristics</b>	Board name	PXI-6508
	Manufacturer	National Instruments
	Bus type	PXI (Compact PCI)
	Access method	I/O mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# National Instruments PXI-6508 Digital Input

**Purpose** PXI-6508 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** The PXI-6508 has four 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling of Input to Output

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

## 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 of 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 either A, B, or C. The I/O board has a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 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.

# National Instruments PXI-6508 Digital Input

---

## **Chip**

From the list choose 1, 2, 3, or 4.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6508 Digital Output

**Purpose** PXI-6508 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** The PXI-6508 has four 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 a 8255 chip with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum of 8 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.

# National Instruments PXI-6508 Digital Output

---

## **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.

## **Chip**

From the list choose 1, 2, 3, or 4.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** National Instruments PXI-6527

## General Description

The PXI-6527 is an I/O board with 24 filtered digital input lines and 24 digital output lines. You can individually turn input filtering on or off for each of the 24 input lines. However, there is only one filter time of all of the input lines that have filtering enabled.

xPC Target supports this board with these driver blocks:

- National Instruments PXI-6527 Digital Input
- National Instruments PXI-6527 Digital Output

## Board Characteristics

Board name	PXI-6527
Manufacturer	National Instruments
Bus type	PXI
Access method	Memory mapped
Multiple block instance support	Digital I/O: Yes
Multiple board support	Yes

# National Instruments PXI-6527 Digital Input

---

**Purpose** PXI-6527 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** The PXI-6527 has one chip with 3 ports (A,B,C) for digital input. Each port has a maximum of 8 digital I/O lines. Use a separate driver block for each port.

---

**Note** The interrupt on change capability of this board is not used.

---

## Scaling of Input to Output

Hardware Input	Block Input Data Type	Scaling
OPTO	Double	$< 0.5 = 0$ $\geq 0.5 = 1$

## 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 of the digital inputs for one port, enter

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

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

### Port

From the list choose either A, B, or C. The **Port** parameter defines which port is used for this driver block. Each port has a maximum of 8 digital inputs. In each case, one block is needed for each port.



# National Instruments PXI-6527 Digital Input

---

The documentation for this board from National Instruments labels these ports 0, 1, and 2.

## **Filter vector**

This is a Boolean vector that selects which input lines to filter. For example, if you enter a **Channel vector** of [1, 3, 5], and you want to filter all three lines, enter

[1, 1, 1]

If you want to filter lines 1 and 5, but not line 3, then enter

[1, 0, 1]

If the filter vector is a single element, then it is scalar expanded to the same width as the **Channel vector**. In the example above, if the **Filter vector** is [0], it is expanded to [0, 0, 0]. Likewise, if the **Filter vector** is [1] it is expanded to [1, 1, 1].

## **Filter interval**

Enter the time interval the hardware filter uses to determine a stable pulse. If the input pulse is shorter than this interval, it is ignored. There is only one filter interval for all filtered inputs, and if you filter on more than one digital input block for this board, you must enter the same **Filter interval** for all blocks.

A reasonable value for the **Filter interval** would be 200 to 300 us.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# National Instruments PXI-6527 Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PXI-6527 Digital Output

**Purpose** PXI-6527 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** The PXI-6527 has one chip with 3 ports (A,B,C) for digital output. Each port has a maximum of 8 digital output lines. Use a separate driver block for each port.

## Scaling of Input to Output

Hardware Output	Block Output Data Type	Scaling
OPTO	Double	<0.5 = OPTO is OFF, no current flowing ≥0.5 = OPTO is ON, current can flow

## 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 of the digital outputs for one port, enter

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

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

### Port

From the list choose either A, B, or C. The **Port** parameter defines which port is used for this driver block. Each port has a maximum of 8 digital output lines. In each case, one block is needed for each port.

# National Instruments PXI-6527 Digital Output

---

The documentation for this board from National Instruments labels these ports 3, 4, and 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 a base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Board** National Instruments PXI-6704

**General Description** The PXI-6704 is an I/O board with 16 voltage outputs, 16 current outputs, and 8 digital I/O lines.

xPC Target supports this board with this driver block:

- National Instruments PXI-6704 Analog Output (D/A)

<b>Board Characteristics</b>	Board name	PXI-6704
	Manufacturer	National Instruments
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	No
	Multiple board support	Yes

# National Instruments PXI-6704 Analog Output (D/A)

**Purpose** PXI-6704 Analog Output block

**Library** xPC Target Library for National Instruments

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 32. Channels 1-16 refer to the voltage output channels, and 17-32 the current output channels. 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.

### **Sample time**

Base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# North Atlantic Industries, Inc. (Formerly Apex)

---

This chapter describes the North Atlantic Industries, Inc. (NAII) I/O boards supported by xPC Target (<http://www.naii.com>).

NAII PC-12SD (PC-77SD1)	I/O board with up to 12 input channels for positioning sensors of type Synchro or Resolver.
NAII (Apex) 73LD3	I/O board with up to 6 input channels for LVDT/RVDT position sensors. The capabilities of this board vary according to its part number designation.
NAII (Apex) 73SD3	I/O board with up to 6 Synchro/Resolver-to-digital channels. The capabilities of this board vary according to its part number designation.
NAII (Apex) 76LD1	I/O board with up to twelve 2-wire or up to six 3-wire or 4-wire stimulus (output) channels for LVDT/RVDT position sensors.
NAII (Apex) 76CL1	I/O board with up to eight 2-wire or up to four 3-wire or 4-wire LVDT/RVDT measurement (input) channels for LVDT/RVDT position sensors.

NAII (Apex) 76CS1

I/O board with up to 8 synchro/resolver measurement (input, S/D) channels and up to 6 stimulus (output, D/S) channels for positioning sensors of type Synchro or Resolver.

“Boards and Blocks —  
Alphabetical List” on page  
27-3

Description of block parameterizeor NAII driver blocks.



## **Boards and Blocks – Alphabetical List**

# NAII PC-12SD (PC-77SD1)

---

**Board** NAII PC-12SD (PC-77SD1)

**General Description** The PC-12SD is an I/O board with up to 12 input channels for positioning sensors of type Synchro or Resolver. The manufacturer individually programs the board according to the order code or board code. Values for some of the block parameters with this driver depend on this board code.

xPC Target supports this board with this driver:

- NAII PC-12SD (PC-77SD1) Synchro/Resolver

<b>Board Characteristics</b>	Board name	PC-12SD
	Manufacturer	NAII
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# NAII PC-12SD (PC-77SD1) Synchro/Resolver

**Purpose** PC-12SD (PC-77SD1) Synchro/Resolver block

**Library** xPC Target Library for NAI

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Synchro or Resolver	Double	angle in rad, velocity in rps

## Block Parameters

### Channel vector

Enter numbers between 1 and 12 to select the Synchro/Resolver (S/D) channels you use with this block. The driver allows the selection of individual S/D channels in any order, but repeating channels is not allowed. For example, to use the first and second S/D channels, enter

[ 1, 2 ]

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

### Sensor type vector (0:resolver, 1:synchro)

If the board code allows switching between Synchro or Resolver inputs, use this vector to define which type of sensor is attached to the corresponding channel. The vector has to have the same length as the **Channel vector**. Use a value 0 to specify Resolver input, and a value 1 to specify Synchro input. If the board code stands for a static type of sensor, enter an empty vector ( [ ] ).

### Ratio vector

Enter numbers between 1 and 255 to define the ratio for each channel. This vector has to have the same length as the **Channel vector**. For single speed input, use a ratio value of 1. For double speed input the ratio depends on your sensor and application and can have a value between 2 and 255. In the case of double speed input, two channels are used to provide fine and coarse data.

# NAII PC-12SD (PC-77SD1) Synchro/Resolver

---

## Output format

From the list, choose Angle, Angle-Status, Angle-Velocity, or Angle-Velocity-Status. This is the output format for each S/D channel and also the format for the output port of this block. The possible selections are:

- Angle — The signal width is 1. This scalar is the angular position in radians.
- Angle - Status — The signal width is 2. The first element is the angular position, and the second element is the status.
- Angle - Velocity — The signal width is 2. The first element is the angular position, and the second element is the angular velocity. The unit for the angular velocity is revolutions/second (rps).
- Angle - Velocity - Status — The signal width is 3. The first element is the angular position, the second element the angular velocity, and the third element is the status.

The status signal returns information about test status, signal status and reference status for each S/D channel. Each status returns binary information (0 is OK, 1 is FAILURE). The test status has weight  $2^0$ , signal status has weight  $2^1$ , and the reference status has weight  $2^2$ .

For example, a status value of 5, means the test status is OK and both signal status and reference status are FAILURE.

Note, if you do not provide a **Reference Vector** by entering an empty matrix, the reference status is not returned. See the board manual for more information about statuses.

## Velocity scaling (max. RPS) vector

Enter a scale factor for defining the maximum rotations/second (rps) for each S/D channel. You need to enter a value to read velocity information. This vector has to have the same length as the **Channel vector**. The values entered here define the maximum revolutions/second and affect the accuracy of the velocity readings. Choose values to have the best accuracy.

## **Show input ports for dynamic velocity scaling**

Selecting this check box allows you to update the **Velocity scaling vector** at runtime.

If checked, the block shows the same number of input ports as output ports. That is, one port for each selected S/D channel. The signal width of each input port is 1. You can use the signal entering the corresponding input port to update the **Velocity scaling vector**. Even if you select this check box and you provide values to the input ports, you still have to enter a **Velocity scaling vector**. In this case, the **Velocity scaling vector** defined the initial values.

## **Reference vector (frequency, voltage)**

If the board code includes the Reference Output option, you can use this vector to define the frequency and amplitude of the reference output. If you enter an empty matrix ([ ]), the reference output circuit is not accessed, even if the board is equipped with it.

To activate the reference output, you have to enter a row vector with two elements, where the first element defines the frequency in Hertz and the second element defines the output voltage in Volts.

## **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 corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# NAII (Apex) 73LD3

---

## Board

NAII (Apex) 73LD3

## General Description

The NAII (Apex) 73LD3 is an I/O board with up to six input channels for LVDT/RVDT position sensors. The capabilities of this board vary according to its part number designation. There can be 2, 4, or 6 channels. The board might contain an on-board reference. If this reference is present, the supported frequency range is board dependent. The acceptable ranges of values for the block parameters vary according to board type.

xPC Target supports this board with this driver:

- NAII 73LD3 LVDT/RVDT Converter

xPC Target does not support the digital I/O functionality of this board.

## Board Characteristics

Board name	73LD3
Manufacturer	NAII
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# NAII 73LD3 LVDT/RVDT Converter

**Purpose** 73LD3 LVDT/RVDT Converter block

**Library** xPC Target Library for NAI

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	LVDT or RVDT	Double	-32768...32767

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first three analog input (A/D) channels, enter

[1, 2, 3]

The channel numbers can occur in any order. Number them beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number can be 2, 4, or 6 depending on the board type.

### **Signal scale vector**

For 2 wire applications, enter the transformation ratio so that the full range of motion results in a full scale output signal from -32768 through 32767. For 3 or 4 wire applications, a setting of 65535 results in a full scale output signal. This scale factor can be chosen for each channel independently. If a scalar value is entered, it is applied to all channels.

### **Velocity scale vector**

Enter the velocity (strokes/s or rev/s) that results in an output signal equal to 32767, the maximum 16-bit integer. This scale factor can be chosen for each channel independently. If a scalar value is entered, it is applied to all channels.

### **Show input ports for dynamic velocity scaling**

Select this check box if you want to be able to update the velocity scale vector at run-time. If this check box is selected, the block

# NAII 73LD3 LVDT/RVDT Converter

---

displays one input port for each output channel selected. These input ports are all of width 1 and can be used to update the velocity scale vector dynamically. Even if you select this check box and provide values to the input ports, you still must enter a velocity scale vector to specify the initial values.

## Output format

From the list select an output format for all channels:

- position — A signal width of 1 that contains the linear position.
- position-status — A signal width of 2 that contains the linear position and status.
- position-velocity — A signal width of 2 the contains the linear position and velocity.
- position-velocity-status — A signal width of 3 that contains the linear position, velocity, and status.

The following notes apply to these formats:

- The position and velocity values are each normalized to be between -1 and 1.
- Position and velocity are returned in radians and radians per second, respectively.
- The status value is encoded in a 3-bit integer that is returned as a double. The status value consists of the following status information for the channel:
  - test status, weight of  $2^0$
  - signal status, weight of  $2^1$
  - reference status, weight of  $2^2$

Each status is a binary value (0 is OK, 1 is FAILURE). The driver encodes these binary values into a single signal. For example, a status value of 5 (100), means the test status is OK and both signal status and reference status are FAILURE.



If the board does not provide a reference/excitation, the test status and reference status have no effect.

When you request a format with status, you also enable the automatic background (bit D2 of the Test Enable Register) testing.

## **Wiring**

From the list, select either 2 wire or 3 or 4 wire.

## **Excitation frequency (Hz)**

Enter the reference frequency of the board in hertz. Note that the possible ranges of reference frequencies depend on the board type and the jumper settings on the board.

## **Excitation voltage (RMS)**

Enter the reference root-mean-square voltage value. Note that the possible ranges of reference voltages depend on the board type.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. Be sure that this corresponds to the actual DIP-switch settings on the board. For example, if the base address is 300 in hexadecimal, enter

0x300

# NAII (Apex) 73SD3

---

**Board** NAII (Apex) 73SD3

**General Description** The NAII (Apex) 73SD3 is an I/O board with up to six Synchro/Resolver-to-digital channels. The capabilities of this board vary according to its part number designation. There can be 2, 4, or 6 channels. The board might contain an on-board reference. If this reference is present, the supported frequency range is board dependent. The acceptable ranges of values for the block parameters vary according to board type.

xPC Target supports this board with one driver block:

- NAII 73SD3 Synchro/Resolver

<b>Board Characteristics</b>	Board name	73SD3
	Manufacturer	NAII
	Bus type	ISA
	Access method	Memory mapped
	Multiple block instance support	No
	Multiple board support	Yes

**Purpose** NAII 73SD3 Synchro/Resolver block

**Library** xPC Target Library for NAII

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Synchro or Resolver	Double	position: radians velocity: radians/second

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the measurement (input) channels. For example, to use the Synchro/Resolver channels 1 and 4, enter

[ 1, 4 ]

The channel numbers can occur in any order. The maximum allowable channel number can be 2, 4, or 6 depending on the board type.

### **Two-speed ratio vector (1: single speed; greater than 1: two-speed)**

Enter a two-speed ratio vector of integers greater than 0. This must be either a scalar or a vector the same length as the channel vector.

The two-speed ratio is normally 1 unless you are using a contiguous odd-even channel pair (such as 3 and 4) as a single two-speed channel. In this case, the two-speed ratio represents the gear ratio between the odd (coarse) and even (fine) channels of the pair. Include only the even member of a two-speed channel pair in the channel vector. For example, if you want to use channels 3 and 4 as a two-speed channel geared in the ratio of 36:1, perform the following:

# NAII 73SD3 Synchro/Resolver

---

- Add the value 4 (but not the value 3) to the **Channel vector** value
- Set the corresponding **Two-speed ratio vector** value to 36

## **High resolution vector (0: 16-bit; 1:24-bit)**

Enter a high resolution vector for the channels. This must be a scalar or a vector the same length as the **Channel vector** value. Enter one of the following values:

- 0 — Specifies a 16-bit channel
- 1 — Specifies a 24-bit channel

Note, you can specify high resolution only for the even-numbered channels of a two-speed channel pair. Specifying high resolution incurs a small additional overhead.

## **Encoder vector (4, 6, 8: commutator poles; 12-16: encoder bits)**

Enter a scalar or a vector that is the same length as the **Channel vector** value. The encoder vector controls the encoder or commutation outputs on connector JP5. Enter values according to the following:

- Commutator — Enter values from the set [4 6 8]. These values specify the number of poles for a commutator output.
- Encoder — Enter values from the set [12 13 14 15 16]. These values specify the number of bits for an encoder output.

## **Output format**

From the list select an output format for all channels:

- position — A signal width of 1 that contains the linear position.
- position-status — A signal width of 2 that contains the linear position and status.
- position-velocity — A signal width of 2 the contains the linear position and velocity.

- position-velocity-status — A signal width of 3 that contains the linear position, velocity, and status.

The following notes apply to these formats:

- The position and velocity values are each normalized to be between -1 and 1.
- Position and velocity are returned in radians and radians per second, respectively.
- The status value is encoded in a 3-bit integer that is returned as a double. The status value consists of the following status information for the channel:
  - Test status, weight of  $2^0$
  - Signal status, weight of  $2^1$
  - Reference status, weight of  $2^2$

Each status is a binary value (0 is OK, 1 is FAILURE). The driver encodes these binary values into a single signal. For example, a status value of 5 (100), means the test status is OK and both signal status and reference status are FAILURE. If the board does not provide a reference/excitation, the test status and reference status have no effect.

When you request a format with status, you also enable the automatic background (bit D2 of the Test Enable Register) testing.

### **Max RPS vector**

Enter the maximum measurable velocity (in revolutions per second) for each channel. This value sets the velocity scale for each channel. This value must be a scalar or a vector that is the same length as the **Channel vector** value. Enter positive numbers less than 152.

### **Show input ports for dynamic max RPS**

Select this check box to display an input port for each channel to be used for specifying the maximum RPS dynamically. When you

# NAII 73SD3 Synchro/Resolver

---

select this check box, the input port signal is internally limited to between 9.5367 and 152.5878 RPS.

## **Latch all channels before reading position data**

Select this check box to guarantee that all channels are latched before being read. This ensures that all channels are sampled at exactly the same time. Note that selecting this option incurs a small additional resource overhead.

## **Excitation frequency (47-10000)**

Enter the frequency of the on-board reference/excitation.

## **Excitation voltage (0, 2-28, or 115)**

Enter the voltage of the on-board reference/excitation.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. Be sure that this corresponds to the actual DIP-switch settings on the board. For example, if the base address is 300 in hexadecimal, enter

0x300

**Board** NAII (Apex) 76LD1

**General Description**

The NAII (Apex) 76LD1 is an I/O board with up to 12 2-wire or up to six 3-wire or 4-wire stimulus (output) channels. The capabilities of this board vary according to its part number designation. The board might contain an on-board reference. If this reference is present, the supported frequency range is board dependent. The acceptable ranges of values for the block parameters vary according to board type.

xPC Target supports this board with one driver block:

- NAII 76LD1 L/D

**Board Characteristics**

Board name	76LD1
Manufacturer	NAII
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# NAII 76LD1 L/D

---

**Purpose** NAI 76LD1 L/D block

**Library** xPC Target Library for NAI

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	LVDT or RVDT	Double	-1...1

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the measurement (input) channels. For example, to use the L/D channels 1 and 3, enter

[1, 3]

The channel numbers can occur in any order. The maximum allowable channel number can be 0, 2, 4, or 8 depending on the board type.

### **Wiring vector (2 for 2-wire;3 for 3-wire or 4-wire)**

Enter the wire type for each of the channels in **Channel vector**. This entry must be either a scalar or a vector that is the same length as the **Channel vector** entry. Specify the wiring vector as follows:

- 2 — Specifies that the corresponding channel is 2-wire
- 3 — Specifies that the corresponding channel is 3-wire or 4-wire

For example, for a **Channel vector** entry of [1, 3], a **Wiring vector** entry of

[2, 3]

specifies that channel 1 is 2-wire and channel 3 is 3-wire or 4-wire.

### **Output format**

From the list select an output format for all channels:



- position — A signal width of 1 that contains the linear position.
- position-status — A signal width of 2 that contains the linear position and status.
- position-velocity — A signal width of 2 the contains the linear position and velocity.
- position-velocity-status — A signal width of 3 that contains the linear position, velocity, and status.

The following notes apply to these formats:

- The position and velocity values are each normalized to be between -1 and 1.
- Position and velocity are returned in radians and radians per second, respectively.
- The status value is encoded in a 3-bit integer that is returned as a double. The status value consists of the following status information for the channel:
  - test status, weight of  $2^0$
  - signal status, weight of  $2^1$
  - reference status, weight of  $2^2$

Each status is a binary value (0 is OK, 1 is FAILURE). The driver encodes these binary values into a single signal. For example, a status value of 5 (100), means the test status is OK and both signal status and reference status are FAILURE. If the board does not provide a reference/excitation, the test status and reference status have no effect.

When you request a format with status, you also enable the automatic background (bit D2 of the Test Enable Register) testing.

**Latch all channels before reading position data**

Select this check box to guarantee that all channels are latched before being read. This ensures that all channels are sampled at exactly the same time. Note that selecting this option incurs a small additional resource overhead.

**Excitation frequency (360-10000)**

Enter the frequency for the on-board reference/excitation signal.

**Excitation voltage (0, 2-28, or 115)**

Enter the voltage of the on-board reference/excitation signal.

**Sample time**

Enter the base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the device ID of the NAI 76LD1 board is 7631 in hexadecimal.

**Board** NAII (Apex) 76CL1

**Library** xPC Target Library for NAII

**Description** The NAII (Apex) 76CL1 I/O board provides up to eight 2-wire or up to four 3-wire or 4-wire LVDT/RVDT measurement (input) channels. It also provides up to 12 2-wire or up to six 3-wire or 4-wire stimulus (output) channels. The capabilities of this board vary according to its part number designation. The board might contain an on-board reference. If this reference is present, the supported frequency range is board dependent. The acceptable ranges of values for the block parameters vary according to board type.

xPC Target supports this board with two driver blocks:

- NAII 76CL1 L/D
- NAII 76CL1 D/L

<b>Board Characteristics</b>	Board name	76CL1
	Manufacturer	NAII
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	No
	Multiple board support	Yes

# NAII 76CL1 L/D

---

**Board** NAII 76CL1 L/D block

**Library** xPC Target Library for NAII

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	LVDT or RVDT	Double	-1...1

## **Block Parameters**

### **Channel vector**

Enter a vector of numbers to specify the stimulus (output) channels. For example, to use the L/D channels 1 and 3, enter

[1, 3]

The channel numbers can occur in any order. The maximum allowable channel number can be 0, 2, 4, or 8 depending on the board type.

### **Wiring vector (2 for 2-wire;3 for 3-wire or 4-wire)**

This must be either a scalar or a vector the same length as the **Channel vector** value. Enter a value of 2 to indicate that the corresponding channel is 2-wire. Enter a value of 3 to indicate the corresponding channel is 3-wire or 4-wire. For example, a **Channel vector** value of [1, 3] and a wiring vector setting of

[2, 3]

indicates that channel 1 is 2-wire and that channel 3 is 3-wire or 4-wire.

### **Output format**

From the list select an output format for all channels:

- **position** — A signal width of 1 that contains the linear position.

- position-status — A signal width of 2 that contains the linear position and status.
- position-velocity — A signal width of 2 the contains the linear position and velocity.
- position-velocity-status — A signal width of 3 that contains the linear position, velocity, and status.

The following notes apply to these formats:

- The position and velocity values are each normalized to be between -1 and 1.
- Position and velocity are returned in radians and radians per second, respectively.
- The status value is encoded in a 3-bit integer that is returned as a double. The status value consists of the following status information for the channel:
  - test status, weight of  $2^0$
  - signal status, weight of  $2^1$
  - reference status, weight of  $2^2$

Each status is a binary value (0 is OK, 1 is FAILURE). The driver encodes these binary values into a single signal. For example, a status value of 5 (100), means the test status is OK and both signal status and reference status are FAILURE. If the board does not provide a reference/excitation, the test status and reference status have no effect.

When you request a format with status, you also enable the automatic background (bit D2 of the Test Enable Register) testing.

### **Latch all channels before reading position data**

Select this check box to guarantee that all channels are latched before being read. This ensures that all channels are sampled at exactly the same time. Note that selecting this option incurs a small additional resource overhead.

# NAII 76CL1 L/D

---

## **Excitation frequency (360-10000)**

Enter the frequency of the on-board reference/excitation.

## **Excitation voltage (0, 2-28, or 115)**

Enter the voltage of the on-board reference/excitation.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the device ID of the NAII 76CL1 board is 7651 in hexadecimal.

**Purpose** NAII 76CL1 D/L block

**Library** xPC Target Library for NAII

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	LVDT/RVDT	Double	-1...1

**Block Parameters**

**Channel vector**

Enter a vector of numbers to specify the stimulus (output) channels. For example, to use the D/L channels 1 and 3, enter

[1, 3]

The channel numbers can occur in any order. The maximum allowable channel number can be 0, 2, 4, or 6 depending on the board type.

If a channel *n* is configured as a 2-wire channel pair (as specified in the **Wiring vector** parameter), enter the A and B subchannels in **Channel vector** as *n* and *-n*, respectively. For example, if all channels are configured as 2-wire, the **Channel vector** value

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

refers to channels 1A, 1B, 2A, 2B, and so on in that order.

**Reset vector**

Enter a scalar or a vector that is the same length as the **Channel vector** value. The reset vector controls the behavior of the channel at model termination. If you specify a scalar value, that setting is used for all channels. If you specify a value of 1, the corresponding channel resets 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**

Enter a scalar or a vector that is the same length as the **Channel vector** value. The initial value vector contains the initial voltage values (in the range -1 to 1) for the output channels. 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.

## **Wiring vector (2 for 2-wire;3 for 3-wire or 4-wire)**

Enter the wire type for each of the channels in **Channel vector**. This entry must be either a scalar or a vector that is the same length as the **Channel vector** entry. Specify the wiring vector as follows:

- 2 — Specifies that the corresponding channel is 2-wire
- 3 — Specifies that the corresponding channel is 3-wire or 4-wire

Note that if a channel is 2-wire, you can specify A and B subchannels (as described for the **Channel vector** parameter).

For example, a **Channel vector** of

[5, 6]

and a **Wiring vector** value of

[2, 3]

indicates that channel 5 is 2-wire and channel 6 is 3-wire or 4-wire. In this channel vector, 5 refers to channel 5A. Channel 6, because it 3-wire or 4-wire, does not have this distinction. The corresponding input port on the block changes its label to 5A when you click **Apply** or **OK**.

On the other hand, the channel vector

[-5, 6]

designates channels 5B and 6.



## **Transformation ratio vector**

Enter the transformation ratio vector for the channels specified in the **Channel vector** value. This entry must be either a scalar or a vector that is the same length as the **Channel vector** value. Enter a ratio within the range 0-2. For 2-wire channels, the transformation ratio applies to both A and B channels.

## **Show output status port**

Select this check box to enable the automatic background (bit D2 of the Test Enable Register) testing. This option displays a status output port labeled **S** when you click **Apply** or **OK**. This port emits a signal with a width of 3. The signal contains the following words for the board:

- test status
- signal status
- excitation status

## **Excitation frequency (360-10000)**

Enter the frequency of the on-board reference/excitation.

## **Excitation voltage (0, 2-28, or 115)**

Enter the voltage of the on-board reference/excitation.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## **NAII 76CL1 D/L**

---

Note that the device ID of the NAII 76CL1 board is 7651 in hexadecimal.

**Board** NAII (Apex) 76CS1

**General  
Block  
Description**

The NAII (Apex) 76CS1 I/O board provides up to 8 synchro/resolver measurement (input, S/D) channels and up to six stimulus (output, D/S) channels. The capabilities of this board vary according to its part number designation. The board might contain an on-board reference. If this reference is present, the supported frequency range is board dependent.

xPC Target supports this board with two driver blocks:

- NAII 76CS1 S/D
- NAII 76CS1 D/S

**Board  
Characteristics**

Board name	76CS1
Manufacturer	NAII
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# NAII 76CS1 S/D

---

**Purpose** NAII 76CS1 S/D block

**Library** xPC Target Library for NAII

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Synchro or Resolver	Double	position: radians velocity: radians/second

**Block Parameters** **Channel vector**  
Enter a vector of numbers to specify the input channels. For example, to use the measurement (S/D) channels 1 and 4, enter

[1, 4]

The channel numbers can occur in any order. The maximum allowable channel number can be 0, 2, 4, or 8 depending on the board type.

**Two-speed ratio vector (1: single speed; greater than 1: two-speed)**

Enter a two-speed ratio vector of integers greater than 0. This must be either a scalar or a vector the same length as the **Channel vector** value.

The two-speed ratio is normally 1 unless you are using a contiguous odd-even channel pair (such as 3 and 4) as a single two-speed channel. In this case, the two-speed ratio represents the gear ratio between the odd (coarse) and even (fine) channels of the pair. Include only the even member of a two-speed channel pair in the **Channel vector**. For example, if you want to use channels 3 and 4 as a two-speed channel geared in the ratio of 36:1, perform the following:

- Add the value 4 (but not the value 3) to the **Channel vector** value
- Set the corresponding **Two-speed ratio vector** value to 36

### **High resolution vector (0: 16-bit; 1:24-bit)**

Enter a high resolution vector for the channels. This must be a scalar or a vector the same length as the **Channel vector** value. Enter one of the following values:

- 0 — Specifies a 16-bit channel
- 1 — Specifies a 24-bit channel

Note, you can specify high resolution only for the even-numbered channels of a two-speed channel pair. Specifying high resolution incurs a small additional overhead.

### **Synchro vector**

Enter a value to indicate that a channel is synchro or resolver. This value must be a scalar or a vector that is the same length as the **Channel vector** value. For each channel, enter

- 1 to indicate that the corresponding channel is a synchro
- 0 to indicate that it is a resolver

### **Encoder vector (4, 6, 8: commutator poles; 12-16: encoder bits)**

Enter a scalar or a vector that is the same length as the **Channel vector** value. The encoder vector controls the encoder or commutation outputs on connector JP5. Enter values according to the following:

- Commutator — Enter values from the set [4 6 8]. These values specify the number of poles for a commutator output.

### **Encoder**

Enter values from the set [12 13 14 15 16]. These values specify the number of bits for an encoder output.

## Output format

- position — A signal width of 1 that contains the linear position.
- position-status — A signal width of 2 that contains the linear position and status.
- position-velocity — A signal width of 2 the contains the linear position and velocity.
- position-velocity-status — A signal width of 3 that contains the linear position, velocity, and status.

From the list select an output format for all channels:

The following notes apply to these formats:

- The position and velocity values are each normalized to be between -1 and 1.
- Position and velocity are returned in radians and radians per second, respectively.
- The status value is encoded in a 3-bit integer that is returned as a double. The status value consists of the following status information for the channel:
  - test status, weight of  $2^0$
  - signal status, weight of  $2^1$
  - reference status, weight of  $2^2$

Each status value is a binary value (0 is OK, 1 is FAILURE). The driver encodes these binary values into a single signal. For example, a status value of 5 (100), means the test status is OK and both signal status and reference status are FAILURE. If the board does not provide a reference/excitation, the test status and reference status have no effect.

When you request a format with status, you also enable the automatic background (bit D2 of the Test Enable Register) testing.

## **Max RPS vector**

Enter the maximum measurable velocity (in revolutions per second) for each channel. This value sets the velocity scale for each channel. This value must be a scalar or a vector that is the same length as the **Channel vector** value. Enter positive numbers less than 152.

## **Show input ports for dynamic max RPS**

Select this check box to display an input port for each channel to be used for specifying the maximum RPS dynamically. When you select this check box, the input port signal is internally limited to between 9.5367 and 152.5878 RPS.

## **Latch all channels before reading position data**

Select this check box to guarantee that all channels are latched before being read. This ensures that all channels are sampled at exactly the same time. Note that selecting this option incurs a small additional resource overhead.

## **Save board setup at model termination (takes 5 seconds)**

Select this check box to save the current board settings to the board at model termination. This action takes approximately 5 seconds. Wait for the save action to complete before removing power from the board. Upon save completion, the message

Setup has been saved

appears in the message window on the upper right of the target PC screen.

## **Excitation frequency (47-1000)**

Enter the frequency of the on-board reference/excitation.

## **Excitation voltage (0, 2-28, or 115)**

Enter the voltage of the on-board reference/excitation.

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## PCI slot

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the device ID of the NAI 76CS1 board is 7621 in hexadecimal.



**Purpose** NAII 76CS1 D/S block

**Library** xPC Target Library for NAII

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Synchro or Resolver	Double	radians

**Block  
Parameters**

**Channel vector**

Enter a vector of numbers to specify the stimulus (output) channels. For example, to use the D/S channels 1 and 3, enter

[1, 3]

The channel numbers can occur in any order. The maximum allowable channel number can be 0, 2, 4, or 6 depending on the board type.

**Reset vector**

Enter a scalar or a vector that is the same length as the **Channel vector** value. The reset vector controls the behavior of the channel at model termination. If you specify a scalar value, that setting is used for all channels. If you specify a value of 1, the corresponding channel resets 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**

Enter a scalar or a vector that is the same length as the channel vector. The initial value vector contains the initial values (in radians) for the output channels. 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.

## **Two-speed ratio vector (1: single speed; greater than 1: two-speed)**

Enter a two-speed ratio vector of integers greater than 0. This must be either a scalar or a vector the same length as the channel vector.

The two-speed ratio is normally 1 unless you are using a contiguous odd-even channel pair (such as 3 and 4) as a single two-speed channel. In this case, the two-speed ratio represents the gear ratio between the odd (coarse) and even (fine) channels of the pair. Include only the even member of a two-speed channel pair in the channel vector. For example, if you want to use channels 3 and 4 as a two-speed channel geared in the ratio of 36:1, perform the following:

- Add the value 4 (but not the value 3) to the **Channel vector** value
- Set the corresponding **Two-speed ratio vector** value to 36

## **High resolution vector (0: 16-bit; 1:24-bit)**

Enter a high resolution vector for the channels. This must be a scalar or a vector the same length as the **Channel vector** value. Enter one of the following values:

- 0 — Specifies a 16-bit channel
- 1 — Specifies a 24-bit channel

Note, you can specify high resolution only for the even-numbered channels of a two-speed channel pair.

Specifying high resolution incurs a small additional overhead.

## **Show output status port**

Select this check box to enable the automatic background (bit D2 of the Test Enable Register) testing. This option displays a status output port labeled **S** when you click **Apply** or **OK**. This port emits a signal with a width of 3. The signal contains the following words for the board:

- test status
- signal status
- excitation status

**Excitation frequency (47-10000)**

Enter the frequency of the on-board reference/excitation.

**Excitation voltage (0, 2-28, or 115)**

Enter the voltage of the on-board reference/excitation.

**Sample time**

Enter the base sample time or a multiple of the base sample time.

**PCI slot**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the device ID of the NAI 76CS1 board is 7621 in hexadecimal.



# Quanser

---

This chapter describes the Quanser I/O board supported by xPC Target (<http://www.quanser.com>).

## Quanser Q8

Multifunction measurement and control board with eight 14-bit single-ended A/D converters, eight 12-bit D/A voltage outputs, 32 digital communication channels, eight 24-bit encoder inputs, and two 32-bit counter/timers.

## **Boards and Blocks**

**Board** Quanser Q8

## General Description

The Quanser Q8 is a multifunction measurement and control board. It has eight 14-bit singled-ended A/D converters (supported by two A/D chips with four channels per chip), eight 12-bit D/A voltage outputs, 32 digital communication channels (each of which can be used for digital input or digital output), eight 24-bit encoder inputs, and two 32-bit counter/timers.

xPC Target supports this board with these driver blocks:

- Quanser Q8 Analog Input
- Quanser Q8 Analog Output
- Quanser Q8 Digital Input
- Quanser Q8 Digital Output
- Quanser Q8 Incremental Encoder
- Quanser Q8 Counter

## Board Characteristics

Board name	Q8
Manufacturer	Quanser
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# Quanser Q8 Analog Input

---

**Purpose** Q8 Analog Input block

**Library** xPC Target Library for Quanser

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter a vector of numbers between 1 and 8. This driver allows you to enter channel numbers in any order. For example, to use the first, second and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if the board manufacturer starts numbering the channels with 0. Note that the input range is -10V to 10V for all channels and is not reconfigurable.

Two A/D chips on the Q8 board convert these channels. One A/D chip converts channels 1 to 4, the second A/D chip converts channels 5 to 8. To maximize throughput, balance the conversion load between the two groups of channels. For example, if you want to convert four channels, a channel vector value like the following has one A/D chip convert channels 1 and 2 and the other A/D chip convert channels 5 and 6:

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

### Sample time

Base sample time or a multiple of the base sample time.



## PCI slot

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of the Quanser Q8 board is 0x0010.

# Quanser Q8 Analog Output

---

**Purpose** Q8 Analog Output block

**Library** xPC Target Library for Quanser

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter a vector of numbers between 1 and 8. This driver allows you to enter channel numbers in any order. For example, to use the first, second and fifth channels, enter

[1,2,5]

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

Two A/D chips on the Q8 board convert these channels. One A/D chip converts channels 1 to 4, the second A/D chip converts channels 5 to 8. To maximize throughput, balance the conversion load between the two groups of channels. For example, if you want to convert four channels, a channel vector value like the following has one A/D chip convert channels 1 and 2 and the other A/D chip convert channels 5 and 6:

[1,2,5,6]

### Range vector

Enter a range code for each of the channels in the channel vector. The value can be a scalar or a vector that must be the same length as the channel vector. If you specify a scalar value, that setting is used for all channels. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 to +10 volts and the second channel is -5 to +5 volts, enter

[ -10, -5]

### Simultaneous update

Choose this check box to ensure that all the outputs update simultaneously (non-transparent mode).

Note that choosing this check box requires an extra register write per sample time and might cause some performance loss.

### 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.

# Quanser Q8 Analog Output

---

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of the Quanser Q8 board is 0x0010.

**Purpose** Q8 Digital Input block

**Library** xPC Target Library for Quanser

**Scaling of Input to Output**

<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL	Double	TTL low=0 TTL high=1

**Block Parameters**

**Channel vector**

Enter a vector of numbers between 1 and 32. This driver allows you to enter channel numbers in any order. For example, to use the first, second and fifth digital channels for input, enter

[ 1,2,5]

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

The Q8 board has 32 digital channels. You can use each of these digital channels for input or output.

---

**Note** If you have a Q8 digital input and digital output block corresponding to the same board (the same PCI slot), do not use the same digital channel number in both blocks.

---

**Sample time**

Base sample time or a multiple of the base sample time.

# Quanser Q8 Digital Input

---

## PCI slot

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of the Quanser Q8 board is 0x0010.

**Purpose** Q8 Digital Output block

**Library** xPC Target Library for Quanser

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter a vector of numbers between 1 and 32. This driver allows you to enter channel numbers in any order. For example, to use the first, second and fifth digital channels for input, enter

[1,2,5]

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

The Q8 board has 32 digital channels. You can use each of these digital channels for input or output.

---

**Note** If you have a Q8 digital input and digital output block corresponding to the same board (the same PCI slot), do not use the same digital channel number in both blocks.

---

### 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.

# Quanser Q8 Digital Output

---

## **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**

Base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of the Quanser Q8 board is 0x0010.



**Purpose** Q8 Incremental Encoder block

**Library** xPC Target Library for Quanser

**Block Parameters** **Channel vector**

Enter a vector of numbers between 1 and 8. This driver allows you to enter channel numbers in any order. For example, to use the first three odd encoder input channels, enter

[ 1,3,5]

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

---

**Note** If you specify four channels or less, it is more efficient to use channels that are all odd or all even. Doing so ensures optimal performance.

---

### **Initial count vector**

The initial value vector contains valid count values to be loaded into the preload register (PR) for the corresponding channel. After a counter decrements to zero, it reloads from the value in the preload register. 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 count value for all channels.

### **Prescale vector**

The prescale vector is a base prescale factor for digital filtering. 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. This filter clock frequency helps eliminate high frequency noise. Enter a value from 0 to 255. An entry of 0 corresponds to the highest possible available frequency of 16.7 MHz.

# Quanser Q8 Incremental Encoder

---

## Quadrature vector

Enter the quadrature vector. 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. The allowable entries are

- 0 — Non-quadrature or normal. The driver treats the A input as count and the B input as direction.
- 1 — 1X quadrature. Counts up or down once per complete cycle of the quadrature signal
- 2 — 2X quadrature. Counts up or down twice per complete cycle of the quadrature signal
- 4 — 4X quadrature. Counts up or down four times per complete cycle of the quadrature signal

## Mode vector

Enter the counting mode. 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. The allowable entries are

- 0 — Normal. The counter wraps from 0 to 0xFFFFFFFF when decrementing and from 0xFFFFFFFF to 0 when incrementing.
- 1 — Range Limit. The counter freezes upon reaching 0 from above and upon reaching the preload register (PR) value from below. In both cases, once counting has stopped, the counter resumes only when the count direction reverses.
- 2 — Non-recycle. Counting stops upon overflow or underflow and does not resume.
- 3 — Modulo-N. This mode is similar to Normal mode except that the maximum count is specified by the preload register (PR).

## Synchronous index vector

Enter the synchronous index vector. 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. The allowable entries are

- 0 — Disables index mode. This mode treats the index/load (I/LD) input as a level-sensitive asynchronous input.
- 1 — Enables index mode. This mode treats the index/load (I/LD) input as a level-sensitive synchronous input with the quadrature clocks.

### **Index polarity vector**

Enter the index polarity mode vector. 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. The allowable entries are

- 0 — Active Low, for a negative slope
- 1 — Active High, for a positive slope

### **Preserve counts vector**

Enter the preserve counts vector. 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. The allowable entries are

- 0 — Do not preserve the current count for the corresponding channel. The driver loads its counter from its preload register (PR) when the model is restarted.
- 1 — Preserve the current count for the corresponding channel when the model is restarted. (Note that the count will be meaningless after power is recycled.)

### **Sample time**

Base sample time or a multiple of the base sample time.

### **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# Quanser Q8 Incremental Encoder

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of the Quanser Q8 board is 0x0010.

**Purpose** Q8 Counter block

**Library** xPC Target Library for Quanser

**Block Parameters**

**Channel vector**

Enter a vector of numbers containing 1 or 2. This driver allows you to enter channel numbers in any order. Channel 1 references the counter channel, channel 2 references the watchdog channel. For example, to use both channels, enter

[1,2]

**Mode vector**

Enter the display port mode. 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. The allowable entries are

- 0 — Frequency Modulation (FM). Display a single input port for the corresponding channel. This port dynamically specifies the lengths (in 30 nanosecond units) of both the low and high phases of a square wave output over the channel.
- 1 — Pulse Width Modulation (PWM). Display a lo and a hi input port for each corresponding channel. These port dynamically specifies the lengths (in 30 nanosecond units) of the low and high phases respectively of a rectangular wave output over the channel.

**Show arm input vector**

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. The allowable entries are

- 0 — Do not display the input arm input. The channel is always armed.
- 1 — Display the input arm input for the associated channel. This setting allows a channel to be armed or disarmed. You can use this port to enable or disable the channel output

dynamically. To enable or disable the channel output, connect a signal to the input arm input. A signal value of 1 enables the channel output, a value of 0 disables the channel output.

When you disarm a channel, its output is continuously high. When you arm a channel, the corresponding input port(s) control the output, as usual.

The value of an input port is undefined for the time between when a model is downloaded and the time it is started. If an input port is armed, its initial value is determined by the corresponding **Initial low count vector** and **Initial high count vector** values:

- If either the **Initial low count vector** or **Initial high count vector** value is nonzero, the channel will be armed between download time and model start time
- If both the **Initial low count vector** or **Initial high count vector** value is zero, the channel will be disarmed.

## **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 low count vector**

This parameter specifies the initial width (the time after model download and prior to model start) in 30 nanosecond intervals of the low portion of the output signal for the corresponding channel. 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.

## **Initial high count vector**

This parameter specifies the initial width (the time after model download and prior to model start) in 30 nanosecond intervals of the high portion of the output signal for the corresponding channel. 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.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI slot**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of the Quanser Q8 board is 0x0010.





# Qatech

---

xPC Target uses Qatech (<http://www.qatech.com>) board drivers as part of its solution for serial communications. These blocks are not described in this chapter. Instead, see Chapter 2, “Serial Communications Support” for details.



# Real Time Devices

---

This chapter describes Real Time Devices I/O boards supported by xPC Target (<http://www.rtdusa.com>).

Real Time Devices DM6420	I/O board with 16 single or 8 differential analog input (A/D) channels, 2 analog output (D/A) channels, 8 independent digital I/O lines, 8 dependent digital I/O lines, and 2 counter/timers.
Real Time Devices DM6430	ISA PC/104 I/O board with 16 single or 8 differential analog input (A/D) channels, 1 analog output (D/A) channel, 16 digital I/O lines, and 2 counter/timers.
Real Time Devices DM6604	ISA PC/104 I/O board with 8 analog output (D/A) channels, and 24 digital I/O lines.
Real Time Devices DM6804	ISA PC/104 I/O board with 24 digital I/O lines and 5 counter/timer channels.
Real Time Devices DM6814	16-bit counting board with 3 channels. This board typically connects to incremental encoders.
Real Time Devices DM6816	ISA PC/104 I/O board with 9 independent PWM channels, 8-bit resolution, 3 16-bit timer/counters, and an on-board 8 MHz clock.

Real Time Devices DM7420

PCI PC/104 I/O board with 16 single or 8 differential analog input (A/D) channels, 8 independent digital I/O lines, 8 dependent digital I/O lines, and 9 counter/timers.

“Boards and Blocks — Alphabetical List” on page 30-3

Description of block parameters for Real Time Devices driver blocks.

## **Boards and Blocks – Alphabetical List**

# Real Time Devices DM6420

---

## Board

Real Time Devices DM6420

## General Description

The DM6420 is an I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 500 kHz, 2 analog output (D/A) channels (12-bit), 8 independent digital I/O lines, 8 dependent digital I/O lines, and 2 counter/timers (16-bit).

xPC Target supports this board with these driver blocks:

- Real Time Devices DM6420 Analog Input (A/D)
- Real Time Devices DM6420 Analog Output (D/A)
- Real Time Devices DM6420 Digital Input
- Real Time Devices DM6420 Digital Output

---

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

---

## Board Characteristics

Board name	DM6420
Manufacturer	Real Time Devices
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Real Time Devices DM6420 Analog Input (A/D)

**Purpose** DM6420 Analog Input block

**Library** xPC Target Library for Real Time Devices

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows the selection of individual A/D channels in any order. The number of elements defines the number of A/D channels used.

For example, to use the first, second and fifth channels, enter

```
[1,2,5]
```

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

### **Input coupling vector**

Enter either 1 (single-ended) or 2 (differential) for each of the channels in the channel vector to choose the coupling code. The coupling vector must be the same length as the channel vector. This driver allows the coupling of each channel to be different.

For example, if the first and second channels are single-ended and the fifth channel is a differential input, enter

```
[1,1,2]
```

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

# Real Time Devices DM6420 Analog Input (A/D)

---

## Range vector

Enter a range code for each of the channels entered in the channel vector. The range vector must be the same length as the channel vector. This driver allows a different range for each channel.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 to + 10 volts and the second and fifth channels are -5 to +5 volts, enter

[ -10, -5, -5]

## Gain vector

Enter 1, 2, 4, or 8 for each of the channels in the channel vector to choose the gain code of that channel. The gain vector must be the same length as the channel vector. This driver allows the gain of each channel to be different.

---

**Note** While this board has programmable input ranges of  $\pm 5$ ,  $\pm 10$  and 0 to 10, this driver sets the input range to +10, and then lets you select different input ranges by choosing different gains.

---

The following table is a list of the ranges for this driver given the gain entered in the gain vector.



# Real Time Devices DM6420 Analog Input (A/D)

---

Gain	Range (V)
1	-10 to 10
2	-5 to +5
4	-2.5 to 2.5
8	-1.25 to 1.25

Notice that by increasing the gain code the voltage range is decreased. The gain divides the input voltage range.

For example, if the first channel has a gain code of 1 (10 volt range) and the second and fifth channels have a gain code of 2 (5 volt range), enter

[1,2,2]

## **Sample time**

Base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Real Time Devices DM6420 Analog Output (D/A)

**Purpose** DM6420 Analog Output block

**Library** xPC Target Library for Real Time Devices

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 8 to select the analog output (D/A) channels used. This driver allows the selection of individual channels in any order. The number of elements defines the number of D/A channels used.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### Range vector

Enter a range code for each of the channels entered in the channel vector. The range vector must be the same length as the channel vector. This driver allows a different range for each channel.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Real Time Devices DM6420 Analog Output (D/A)

---

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 and fifth channels are 0 to +5 volts, enter

[10,5,5]

### Sample time

Enter the model base sample time or a multiple of the base sample time.

### Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Real Time Devices DM6420 Digital Input

---

**Purpose** DM6420 Digital Input block

**Library** xPC Target Library for Quanser

**Note** The DAS1601/16 has a 8255 chip with 3 ports (A,B,C). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either A, B, or C. The port name defines which port is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs. In each case, one block is needed for each port.

# Real Time Devices DM6420 Digital Input

---

## **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

# Real Time Devices DM6420 Digital Output

---

**Purpose** DM6420 Digital Output block

**Library** xPC Target Library for Quanser

**Note** The DAS1601/16 has an 8255 chip with 2 ports (1,2). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 1 or 2. The port name defines which port is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as outputs. In each case, one block is needed for each port.

# Real Time Devices DM6420 Digital Output

---

## **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

# Real Time Devices DM6430

---

## Board

Real Time Devices DM6430

## General Description

The DM6430 is an ISA PC/104 I/O board with 16 single or 8 differential analog input (A/D) channels (16-bit) with a maximum sample rate of 100 kHz, 1 analog output (D/A) channel (16-bit), 16 digital I/O lines, and 2 counter/timers (16-bit).

xPC Target supports this board with these driver blocks:

- Real Time Devices DM6430 Analog Input (A/D)
- Real Time Devices DM6430 Analog Output (D/A)
- Real Time Devices DM6430 Digital Input
- Real Time Devices DM6430 Digital Output

---

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

---

## Board Characteristics

Board name	DM6430
Manufacturer	Real Time Devices
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes



# Real Time Devices DM6430 Analog Input (A/D)

**Purpose** DM6430 Analog Input block

**Library** xPC Target Library for Quanser

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows the selection of individual A/D channels in any order. The number of elements defines the number of A/D channels used.

For example, to use the first, second and fifth channels, enter

[ 1,2,5]

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

### **Gain vector**

Enter 1, 2, 4, or 8 for each of the channels in the channel vector to choose the gain code of that channel. The gain vector must be the same length as the channel vector. This driver allows the gain of each channel to be different.

The following table is a list of the ranges for this driver given the gain entered in the gain vector.

<b>Gain</b>	<b>Range (V)</b>
1	-10 to 10
2	-5 to +5

# Real Time Devices DM6430 Analog Input (A/D)

---

Gain	Range (V)
4	-2.5 to 2.5
8	-1.25 to 1.25

Notice that by increasing the gain code the voltage range is decreased. The gain divides the input voltage range.

For example, if the first channel has a gain code of 1 (10 volt range) and the second and fifth channels have a gain code of 2 (5 volt range), enter

[1,2,2]

## Input coupling vector

Enter either 1 (single-ended) or 2 (differential) for each of the channels in the channel vector to choose the coupling code. The coupling vector must be the same length as the channel vector. This driver allows the coupling of each channel to be different.

For example, if the first and second channels are single-ended and the fifth channel is a differential input, enter

[0,0,1]

The driver selects a second differential input 8 channels higher than the first channel. In the example above, the driver would select the thirteenth channel as a differential input with the fifth channel.

## Sample time

Base sample time of a multiple of the base sample time.

# Real Time Devices DM6430 Analog Input (A/D)

---

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Real Time Devices DM6430 Analog Output (D/A)

---

**Purpose** DM6430 Analog Output block

**Library** xPC Target Library for Quanser

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

**Block Parameters** This board has 2 analog outputs (D/A) with a fixed range of -10 to +10 volts.

## **Channel vector**

Enter 1 or 2 to select the digital output 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 of the digital outputs for one port, enter

[1,2]

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

## **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

# Real Time Devices DM6430 Analog Output (D/A)

---

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**

Base sample time of a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Real Time Devices DM6430 Digital Input

---

**Purpose** DM6430 Digital Input block

**Library** xPC Target Library for Quanser

**Note** The DM6430 has an 8255 chip with 2 ports (1,2). Each port has a maximum of 8 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, the port is configured as input.

## Scaling Input to Output

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

## 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 of 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 either 1 or 2. The port name defines which port is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs. In each case, one block is needed for each port.

# Real Time Devices DM6430 Digital Input

---

## **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

# Real Time Devices DM6430 Digital Output

---

**Purpose** DM6430 Digital Output

**Library** xPC Target Library for Quanser

**Note** The DM6430 has a 8255 chip with 2 ports (1,2). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## 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 of 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 1 or 2. The port name defines which port is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as outputs. In each case, one block is needed for each port.



# Real Time Devices DM6430 Digital Output

---

## **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

# Real Time Devices DM6604

---

## Board

Real Time Devices DM6604

## General Description

The DM6604 is an ISA PC/104 I/O board with 8 analog output (D/A) channels (12-bit), and 24 digital I/O lines.

xPC Target supports this board with these driver blocks:

- Real Time Devices DM6604 Analog Output (D/A)
- Real Time Devices DM6604 Digital Input
- Real Time Devices DM6604 Digital Output

## Board Characteristics

Board name	DM6604
Manufacturer	Real Time Devices
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Real Time Devices DM6604 Analog Output (D/A)

**Purpose** DM6604 Analog Output block

**Library** xPC Target Library for Quanser

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter numbers between 1 and 8 to select the analog output (D/A) channels used. This driver allows the selection of individual channels in any order. The number of elements defines the number of D/A channels used.

For example, to use the first, second, and fifth channels, enter

[ 1,2,5]

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

### Range vector

Enter a range code for each of the channels entered in the channel vector. The range vector must be the same length as the channel vector. This driver allows a different range for each channel.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Real Time Devices DM6604 Analog Output (D/A)

---

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

For example, if the first channel is -10 to +10 volts and the second and fifth channels are 0 to +5 volts, enter

[ -10,5,5]

### Sample time

Enter the model base sample time or a multiple of the base sample time.

### Base address

Enter the base address of the board. It is important that this entry corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

**Purpose** DM6604 Digital Input block

**Library** xPC Target Library for Quanser

**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 of 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 either A, B, or C. The port name defines which port is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs. 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

# Real Time Devices DM6604 Digital Output

---

**Purpose** DM6604 Digital Output block

**Library** xPC Target Library for Quanser

## **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 of 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 port name defines which port is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as outputs. 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.

# Real Time Devices DM6604 Digital Output

---

## **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

# Real Time Devices DM6804

---

## Board

Real Time Devices DM6804

## General Description

The DM6804 is an ISA PC/104 I/O board with 24 digital I/O lines and 5 counter/timer channels (16-bit).

It contains one 8255 chip with 3 digital I/O ports and one AM9513A counter/timer chip. For additional information about the various counter/timer modes of that chip see the AM9513A data sheet which is part of the board documentation.

xPC Target supports this board with these driver blocks:

- Real Time Devices DM6804 Digital Input
- Real Time Devices DM6804 Digital Output
- Real Time Devices DM6804 Counter PWM
- Real Time Devices DM6804 Counter PWM & ARM
- Real Time Devices DM6804 Counter FM
- Real Time Devices DM6804 Counter FM & ARM
- Real Time Devices DM6804 PWM Capture
- Real Time Devices DM6804 FM Capture
- Real Time Devices DM6804xx

## Board Characteristics

Board name	DM6804
Manufacturer	Real Time Devices
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes



# Real Time Devices DM6804 Digital Input

---

**Purpose** DM6804 Digital Input block

**Library** xPC Target Library for Quanser

**Note** The DM6804 has a 8255 chip with 3 ports (A, B, C). Each port has a maximum of 8 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, the port is configured as input.

## 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 of 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 either A, B, or C. The port name defines which port is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as inputs. 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
```

# Real Time Devices DM6804 Digital Output

---

**Purpose** DM6804 Digital Output block

**Library** xPC Target Library for Quanser

**Note** The DM6804 has a 8255 chip with 3 ports (A, B, C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## **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 of 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 port name defines which port is used for this driver block. Each port has a maximum of 8 digital lines that can be configured as outputs. 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.

# Real Time Devices DM6804 Digital Output

---

## **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

# Real Time Devices DM6804 Counter PWM

---

**Purpose** DM6804 Counter PWM block

**Library** xPC Target Library for Quanser

**Note** The DM6804 has one AM9513A chip with 5 counters.

The DM6804 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 which 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

## 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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### Relative output frequency

Enter a value between 0 and  $.25 \times \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 87.5 kHz, then choose F2=500kHz as the Frequency Base and enter 0.175 as the Relative Output Frequency.  $500 \text{ kHz} \times 0.175 = 87.5 \text{ kHz}$

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and gets 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

# Real Time Devices DM6804 Counter PWM & ARM

---

**Purpose** DM6804 Counter PWM & ARM block

**Library** xPC Target Library for Quanser

**Note** The DM6804 has one AM9513A chip with 5 counters.

The DM6804 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 to arm and disarm the counter by 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

## 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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### Relative output frequency

Enter a value between 0 and  $.25 \cdot \text{Frequency base}$ . The **Relative output frequency** is multiplied by the **Frequency base** to set the fixed output frequency of the PWM-signal.

For example, if the output frequency of a square wave has to be 87.5 kHz, then choose F2=500kHz as the Frequency Base and enter 0.175 as the Relative Output Frequency.  $500 \text{ kHz} \times 0.175 = 87.5 \text{ kHz}$

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

## **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running. If the application is running, the second input port controls whether the counter is armed or disarmed. This parameter sets the TTL level when the counter is disarmed.

## **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

## **Base address**

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

0x300

# Real Time Devices DM6804 Counter FM

---

**Purpose** DM6804 Counter FM block

**Library** xPC Target Library for Quanser

**Note** The DM6804 has one AM9513A chip with 5 counters.

The DM6804 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 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

## 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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### Output duty cycle

Enter a value between 0 and 1 to set the duty cycle of the square wave. The Duty Cycle is held fixed during execution of the target application.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

### Level sequence of square wave

From the list, choose either high-low or low-high:



- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the duty cycle entering the block defines the duration of the TTL high part.

### **Level when disarmed**

From the list, choose either high or low. The counter is automatically disarmed when the target application is not running and gets 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

# Real Time Devices DM6804 Counter FM & ARM

---

**Purpose** DM6804 Counter FM & ARM block

**Library** xPC Target Library for Quanser

**Note** The DM6804 has one AM9513A chip with 5 counters.

The DM6804 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 to arm and disarm the counter by 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	Variable frequency: Double Arm: Double	<0.5 disarmed ≥0.5 armed

## 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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency. The XTAL frequency is assumed to be 1 MHz, therefore the jumper on the DM6804 has to 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.

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-9 for details.

---

## **Level sequence of square wave**

From the list, choose either high-low or low-high:

- If you choose high-low, the square wave period starts with the TTL high part followed by the TTL low part.
- If you choose low-high, the square wave period starts with the TTL low part followed by the TTL high part.

In either case, the **Output duty cycle** defined in the setting above define 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 corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Real Time Devices DM6804 PWM Capture

---

**Purpose** DM6804 PWM Capture block

**Library** xPC Target Library for Quanser

**Note** This block programs the AMD9513A for capturing PWM signals by using two counters. One counter measures the cycle duration, and the other counter measures the duration 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 has to enter the pins named GATE of both corresponding counter channels (parallel wiring). Both CLK pins have to be left unconnected.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	0 to 1

## Block Parameters

### Counter

From the list, choose 1&2, 2&3, 3&4, 4&5. This selects which two counters the driver block uses to determine the PWM. In each case, one block is needed for each counter.

### Frequency base

From the list, choose F1=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency. The XTAL frequency is assumed to be 1 MHz, therefore the jumper on the DM6804 has to 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)

# Real Time Devices DM6804 PWM Capture

---

## **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

# Real Time Devices DM6804 FM Capture

---

**Purpose** DM6804 FM Capture block

**Library** xPC Target Library for Quanser

**Note** This block programs the AMD9513A 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 has to enter the pin named GATE of the corresponding counter channel. The CLK pin has to be left unconnected.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	0 to 1

## **Block Parameters**

### **Counter**

From the list, choose 1, 2, 3, 4, or 5. 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=5MHz, F2=500kHz, F3=50kHz, F4=5kHz, or F5=500Hz to set the base frequency.

### **Sample time**

Enter the base sample time or a multiple of the base sample time. The sample time indicates the update rate of registration on the input (Duty Cycle)

### **Base address**

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

0x300

**Purpose** DM6804xx block

**Library** xPC Target Library for Quanser

**Description** You can use this block to program the AMD9513A 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.

# Real Time Devices DM6814

---

## Board

Real Time Devices DM6814

## General Description

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 also three I/O ports, with each port containing eight digital I/O lines.

xPC Target supports this board with these driver blocks:

- Real Time Devices DM6814 Incremental Encoder
- Real Time Devices DM6814 Digital Input
- Real Time Devices DM6814 Digital Output

---

**Note** xPC Target 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



# Real Time Devices DM6814 Incremental Encoder

**Purpose** DM6814 Incremental Encoder block

**Library** xPC Target Library for Quanser

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	counts

## **Block Parameters**

### **Encode 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.

For this purpose you must connect the incremental encoder index output to the correct pin: P0.2 for encoder channel 1, P2.2 for channel 2, and P4.2 for channel 3.

### **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

# Real Time Devices DM6814 Digital Input

---

**Purpose** DM6814 Digital Input block

**Library** xPC Target Library for Quanser

**Note** You can use only one digital input block per port. You can use an input and an output block for the same port, but then each block must use channels different from those in the other block.

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

## **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.

# Real Time Devices DM6814 Digital Input

---

## **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

# Real Time Devices DM6814 Digital Output

---

**Purpose** DM6814 Digital Output block

**Library** xPC Target Library for Quanser

**Note** You can configure two digital I/O lines 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 then each block must use channels different from those in the other block.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	<0.5 = TTL low ≥0.5 = TTL high

## **Block Parameters**

### **Channel Vector**

Enter either numbers 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

# Real Time Devices DM6814 Digital Output

---

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

# Real Time Devices DM6816

---

**Board** Real Time Devices DM6816

**General Description** The DM6816 is an ISA PC/104 I/O board with 9 independent PWM channels, 8-bit resolution, 3 16-bit timer/counters, and an on-board 8 MHZ clock.

xPC Target supports this board with this driver block:

- Real Time Devices DM6816 PWM

<b>Board Characteristics</b>	Board name	DM6816
	Manufacturer	Real Time Devices
	Bus type	PC/104
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

**Purpose** DM6816 PWM block

**Library** xPC Target Library for Quanser

**Note** As input to the DM6816 PWM block, use a number ranging from 0 to 255. This range corresponds to the duty cycle, where 256 is a duty cycle of 100 percent and 0 is 0 percent. The maximum duty cycle that can be given as an input is  $255/256 = 0.996$ .

For example, to get a duty cycle of 0.06, input  $0.06 * 256 = 15.36$  as the input to the channel. The nearest integer will be used such that the duty cycle will actually be  $15/256 = 0.05859$ .

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 9 to select the PWM channel. This driver allows the selection of individual PWM channels in any order. The number of elements defines the number of channels used.

For example, to use all of the channels, enter

[1,2,3,4,5,6,7,8,9]

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

### **Clock source for channels 1, 2, 3**

From the list choose either 8 MHz clock or Timer 1 output as the clock source for the channels. By default, it is Timer 1 output. This parameter affects channels 1, 2, or 3.

### **Clock source for channels 4, 5, 6**

From the list choose either 8 MHz clock or Timer 1 output as the clock source for the channels. By default, it is Timer 1 output. This parameter affects channels 4, 5, or 6.

# Real Time Devices DM6816 PWM

---

## **Clock source for channels 7, 8, 9**

From the list choose either 8 MHz clock or Timer 1 output as the clock source for the channels. By default, it is Timer 1 output. This parameter affects channels 7, 8, or 9.

## **Frequency divisors for timers 0, 1, 2**

Enter a vector [d0 d1 d2] of integers, with each integer in the range from 2 to 65535. The driver uses these integers as frequency divisors for the timers 0, 1, and 2, respectively. For example, if timer 0 uses the 8 MHz clock as a source, a frequency divisor value of 4 for d0 causes timer 0 to run at 2 MHz (8 MHz/4).

If you specify one integer in the vector, that value applies to all timers.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **Base address (e.g. 0xd000)**

Enter the base address of the board. It is important that this entry corresponds to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



## Board

Real Time Devices DM7420

## General Description

The DM7420 is a PCI PC/104 I/O board with 16 single or 8 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 600 kHz, 8 independent digital I/O lines, 8 dependent digital I/O lines, and 9 counter/timers.

xPC Target supports this board with these driver blocks:

- Real Time Devices DM7420 Analog Input (A/D)
- Real Time Devices DM7420 Digital Input
- Real Time Devices DM7420 Digital Output

---

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

---

## Board Characteristics

Board name	DM6604
Manufacturer	Real Time Devices
Bus type	PCI (PC104)
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Real Time Devices DM7420 Analog Input (A/D)

**Purpose** DM7420 Analog Input block

**Library** xPC Target Library for Quanser

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

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16. This driver allows the selection of individual A/D channels in any order. The number of elements defines the number of A/D channels used.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

### **Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Real Time Devices DM7420 Analog Input (A/D)

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

[-10,10]

## Gain vector

Enter 1, 2, 4, 8, 16, or 32 for each of the channels in the channel vector to choose the gain code of that channel. The gain vector must be the same length as the channel vector. This driver allows the gain of each channel to be different.

The following table is a list of the ranges for this driver given the gain entered in the gain vector.

Gain	Range (V)
1	0 to +10
2	0 to +5
4	0 to +2.5
8	0 to +1.25
16	0 to +0.625
32	0 to +0.312

Notice that by increasing the gain code the voltage range is decreased. The gain divides the input voltage range.

For example, if the first channel has a gain code of 1 (10 volt range) and the second and fifth channels have a gain code of 2 (5 volt range), enter

[1,2,2]

## Input coupling vector

Enter either 1 (ground-referenced single-ended (RSE)), 2 (nonreferenced singled-ended (NRSE)), or 3 (differential (DIFF)) for each of the channels in the channel vector to choose the

# Real Time Devices DM7420 Analog Input (A/D)

coupling code. The coupling vector must be the same length as the channel vector. This driver allows the coupling of each channel to be different.

The following table is a list of the couplings for this driver and the corresponding coupling codes.

<b>Coupling</b>	<b>Coupling Code</b>	<b>Description</b>
RSE	1	High side of input signal connected to the selected analog input channel. Low side of input signal connected to one of the analog ground (ANALOG GND) pins. See the board manual.
NRSE	2	High side of input signal connected to the selected analog input channel. Low side of the input signal connected to the analog input sense (AINSENSE) pin at the I/O connector. See the board manual.
DIFF	3	High side of input signal connected to the selected analog input channel AIN1+...AIN8+. Low side of input signal connected to AIN- pin corresponding to the AIN+. See the board manual.

For example, if the first and second channels are ground referenced single-ended and the third channel is a differential input, enter

[1,1,3]

For differential inputs, the driver selects a second differential input that is eight channels higher than the first channel. In

# Real Time Devices DM7420 Analog Input (A/D)

---

the example above, the driver selects the eleventh channel (AIN11/AIN3-) as a differential input.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Real Time Devices DM7420 Digital Input

---

**Purpose** DM7420 Digital Input block

**Library** xPC Target Library for Quanser

## **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 of 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 either 0 or 1

### **Sample time**

Base sample time of a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Real Time Devices DM7420 Digital Output

---

**Purpose** DM7420 Digital Output block

**Library** xPC Target Library for Quanser

**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 of 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 0 or 1.

**Sample time**

Base sample time of a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```





# SBS Technologies

---

This chapter describes SBS Technologies I/O boards supported by xPC Target (<http://www.sbs.com>).

“Broadcast Memory” on page 31-3	PCI broadcast memory node.
“SBS Shared Memory Structure Reference” on page 31-8	Shared memory reference for structures.
SBS 2510/2500 Broadcast Memory	High-speed fiber optic reflective memory.
SBS Flex/104A PC/104 IP Carrier Board	PC/104 format IP carrier board with a capacity of 2 IP modules.
SBS IP-16ADC	16 single or 8 differential analog input (A/D) channels (16-bit) with a combined throughput of 100 kHz.
SBS IP-16ADC	3 independent precalibrated analog output (D/A) channels (16-bit).
SBS IP-DAC	6 independent precalibrated analog output (D/A) channels (12-bit).
SBS IP-Digital 24	24 digital I/O lines which can be independently configured for input or output.
SBS IP-HiADC	16 analog input (A/D) channels (12-bit).
SBS IP-Synchro	Two channels of position measurement using synchro, resolver, LVDT, or Inductosyn <sup>®</sup> transducers.
SBS IP-UD-IHV-16I80	IndustryPack board with 16 inputs and 8 outputs.

SBS IP-UD-IHV-8I160	IndustryPack board with eight inputs and 16 outputs.
SBS P-Unidig-E-48	48 digital I/O lines which can be independently configured for input or output.
SBS PCI-40A Carrier Board	ISA format IP carrier board with a capacity of 4 IP modules.
SBS PCI-40B Carrier Board	ISA format IP carrier board with a capacity of 4 IP modules.
“Boards and Blocks — Alphabetical List” on page 31-14	Description of block parameters for SBS Technologies driver blocks.

## Broadcast Memory

xPC Target uses a model for shared memory that includes Simulink blocks, for the shared memory drivers, and MATLAB structures for defining shared memory and node initialization partitions. The topics in this section are

- “Create Shared Memory Partitions” on page 31-3 — Before you begin to use SBS Broadcast Memory shared memory blocks, define a partition structure that defines how you want to allocate shared memory.
- “Initialize Shared Memory Nodes” on page 31-5 — Before you begin to use SBS Broadcast Memory shared memory blocks, define a node initialization structure that defines how shared memory is allocated (partitioned) and how the board is configured.

---

**Note** The xPC Target SBS25x0 driver does not fully support node-to-node network configuration. In most cases, a shared memory hub is required to use the SBS Shared driver blocks. If you have additional questions, please contact The MathWorks Technical Support.

---

### Create Shared Memory Partitions

To use the xPC Target SBS Broadcast Memory shared memory blocks to read, write, pack, or unpack data, you must define a partition structure. SBS Broadcast Memory shared memory drivers use MATLAB structures to define shared memory partitions. A partition structure describes how you want to allocate (or partition) the shared memory. xPC Target allocates shared memory with bundles of data that are packed into memory partitions. The following SBS Broadcast Memory blocks use shared memory partition structures:

- SBS25x0 read and SBS25x0 write
- Shared Memory Pack and Shared Memory Unpack

After you define the shared memory partitions, you can add SBS Broadcast Memory shared memory driver blocks to your Simulink model. Create a shared memory structure in one of the following ways. See “Shared Memory

Partition Structure” on page 31-8 for the complete list of fields for the structure.

The following description refers to the `smpartsbs25x0` command. Type

```
help smpartsbs25x0
```

for a description of the command.

- Using the `smpartsbs25x0` command at the MATLAB Command Window, create a default partition structure. For example, type

```
x=smpartsbs25x0

>> smpartsbs25x0
SBS25x0 Shared Memory Partition
Shared-Memory partition
  Total Bytes in partition = 4
  Starting Address = 0x0
  Number of segments = 1
```

To get the contents of `x`, type

```
>> get(x)

ans =
  Address: '0x0'
  Type: 'uint32'
  Alignment: '4'
  Size: '[ 1 ]'
  WIT: 'off'
```

- At the MATLAB Command Window, create a user-defined partition structure. The easiest way to do this is to create an M-file, partially define a structure, load that M-file into the MATLAB workspace. For example

```
Partition(1).Address='0x5000';
Partition(1).Type='int8';
Partition(1).Size='10';
Partition(2).Type='uint16';
Partition(2).Size='5';
```

```

Partition(3).Type='uint8';
Partition(3).Size='1';
Partition(3).Alignment='8';
Partition(4).Type='double';
Partition(4).Size='3';

```

To get the contents of x with these settings, type

```

>> x=smpartsbs25x0(Partition)
SBS25x0 Shared Memory Partition
Shared-Memory partition
Total Bytes in partition = 16
Starting Address = 0x5000
Number of segments = 4

```

This example defines a partition with four segments.

- The Address field is optional. Only specify this field for the first segment of a partition. The elements of a partition are defined as a continuous memory block from the first address. The following segments extrapolate their addresses from the first segment. If you have fragmented memory, use multiple partitions and SBS Broadcast Memory read/write blocks to work with the memory.
- The Type and Size fields are required for all fields in the partition structure.
- The Alignment value is optional. It is '4' by default, which forces segments that do not have alignment specifications to start on 4 byte (32 bit) boundaries. In this partition, the third segment (Partition(3)) has an alignment of '8'.
- The base addresses of subsequent segments are fully defined by the data type, size, and alignment of the preceding segment.

You can then call the `smpartsbs25x0()` command to fully populate the partition structure.

## Initialize Shared Memory Nodes

To use the xPC Target SBS Broadcast Memory shared memory, you must define a node initialization structure. SBS Broadcast Memory shared

memory drivers use MATLAB structures to define shared memory node initialization. A node initialization structure describes the shared memory partition (see “Create Shared Memory Partitions” on page 31-3) and the SBS Broadcast Memory board configuration, including interrupt configurations. The following SBS Broadcast Memory block requires shared memory node initialization structures:

- SBS25x0 init

After you define the node initialization partition, you can add SBS Broadcast Memory shared memory driver blocks to your Simulink model. Create a shared memory structure in one of the following ways. See “Shared Memory Node Initialization Structure” on page 31-10 for the complete list of fields for the structure.

The following description refers to the `smnodesbs25x0` command. Type

```
help smnodesbs25x0
```

for a description of the command.

Using the `smnodesbs25x0` command at the MATLAB Command Window, create a default node initialization structure. For example, type

```
>> node=smnodesbs25x0
Node Definition for SBS2500/SBS2510 Broadcast Memory
```

To get the contents of `node`, type

```
>> get(node)

ans =
    TargetAbortEnable: 'off'
      LoopbackEnable: 'off'
         Node2Node: 'off'
           RXEnable: 'on'
           TXEnable: 'on'
      MemorySize: '256kByte'
         WITEnable: 'off'
    IRQ_ErrorEnable: 'off'
      IRQ_WITEnable: 'off'
```

```
Partition: []  
SlotID: 'any'
```

- At the MATLAB Command Window, create a user-defined node structure. The easiest way to do this is to create an M-file, partially define a structure, load that M-file into the MATLAB workspace. For example

```
z.LoopbackEnable='on';  
z.Partition=x
```

To get the contents of node with these settings, type

```
>> node=smnodesbs25x0(node1)  
Node Definition for SBS2500/SBS2510 Broadcast Memory  
>> get(node)
```

```
ans =  
TargetAbortEnable: 'off'  
LoopbackEnable: 'on'  
Node2Node: 'off'  
RXEnable: 'on'  
TXEnable: 'on'  
MemorySize: '256kByte'  
WITEnable: 'off'  
IRQ_ErrorEnable: 'off'  
IRQ_WITEnable: 'off'  
Partition: [1x1 smpartsbs25x0]  
SlotID: 'any'
```

Currently, you can define values only for the fields **LoopbackEnable**, **WITEnable**, and **Partition**.

## SBS Shared Memory Structure Reference

You do not need to use all the fields of a partition or node initialization structure. However, knowing the possible structure fields will be helpful when you are creating any of the structures.

This section includes the following topics:

- “Shared Memory Partition Structure” on page 31-8 — Description of the partition node structure fields
- “Shared Memory Node Initialization Structure” on page 31-10 — Description of the node initialization structure fields

### Shared Memory Partition Structure

A shared memory partition structure has the following fields:

```
Address: '0x0'
        Type: 'uint32'
Alignment: '4'
        Size: '[ 1 ]'
        WIT: 'off'
```

where

Partition Fields	Description
<b>Address</b>	Specifies the base address (in hexadecimal) of the memory partition within the node’s shared memory space. The default value is '0x0', the first location in shared memory.  Align partition addresses on 32-bit word boundaries (for example, 0x0, 0x4, 0x8, and so forth).



<b>Partition Fields</b>	<b>Description</b>
<b>Type</b>	<p>Specifies the data type of the memory segment. Specify one of the following types:</p> <ul style="list-style-type: none"> <li>• <code>single</code> (IEEE Single Precision)</li> <li>• <code>double</code> (IEEE Double Precision)</li> <li>• <code>uint8</code></li> <li>• <code>int8</code></li> <li>• <code>uint16</code></li> <li>• <code>int16</code></li> <li>• <code>uint32</code></li> <li>• <code>int32</code></li> <li>• <code>Boolean</code> (a single byte represents a boolean value)</li> </ul> <p>The default value is '<code>uint32</code>'. A minimum partition size is 32 bits.</p>
<b>Size</b>	<p>Specifies the dimension and size of the memory segment. You can enter a scalar value or a value with the <code>[m,n]</code> format. The default value is '<code>1</code>'.</p> <ul style="list-style-type: none"> <li>• <code>scalar</code> — Treats the <code>Size</code> entry as the specification of the length of a non-oriented array or vector</li> <li>• <code>[m,n]</code> — Treats the <code>Size</code> entry as an array dimension. The total number of elements in this segment is <math>m*n</math>.</li> </ul>

Partition Fields	Description
<b>Alignment</b>	If another partition precedes this partition, defines the byte alignment of this segment. Specify one of the following alignment values: 1, 2, 3, 4, or 8. The default value is '4'. This value forces a double word boundary alignment for the first address of this segment.
<b>WIT</b>	<p>Write Interrupt Table. If this partition segment receives a write request, specifies whether or not the partition generates an interrupt. Specify 'off', 'first', 'all', 'last'. The default value is 'off'.</p> <p>'off' — Prevents the partition from generating interrupts</p> <p>'first' — Allows a write to the first double word of the memory block to generate an interrupt.</p> <p>'all' — Allows a write to all locations of the memory block to generate an interrupt.</p> <p>'last' — Allows a write to only the last double word of the memory segment to generate an interrupt.</p>

## Shared Memory Node Initialization Structure

A node initialization structure has the following fields:

```

TargetAbortEnable: 'off'
  LoopbackEnable: 'off'
    Node2Node: 'off'
      RXEnable: 'on'
        TXEnable: 'on'
          MemorySize: '256kByte'
            WITEnable: 'off'
              IRQ_ErrorEnable: 'off'
                IRQ_WITEnable: 'off'
                  Partition: []
                    SlotID: 'any'

```

These values affect the Extended Control Register. Refer to the SBS Broadcast Memory product documentation for further details on registers.

Of particular note are the following register modes:

<b>Board Modes</b>	<b>Description</b>
<b>Loopback Enable</b>	<p>Specifies if xPC Target can operate without having the shared memory network fully operational.</p> <ul style="list-style-type: none"> <li>• 'off' — Normal shared memory operation (requires operating shared memory network)</li> <li>• 'on' — Loop back enable. This allows xPC Target to operate without having the shared memory network fully operational.</li> </ul>
<b>Memory Size</b>	<p>Specifies the minimum memory size required. The SBS Broadcast Memory driver checks this value against the memory size of the SBS Broadcast Memory board. An SBS Broadcast Memory board has a memory size of either '64MByte' or '128MByte'. If you enter a size in this field that is larger than the actual SBS Broadcast Memory board memory size, the driver will return an error. You can enter values of:</p> <p>'256kByte'</p> <p>'512kByte'</p> <p>'1MByte'</p> <p>'2MByte'</p> <p>'4MByte'</p> <p>'8MByte'</p>
<b>Partition</b>	<p>Specifies the memory partition to be initialized. The node requires this value to initialize the interrupt table for the partition. This parameter is required for xPC Target configurations that will be driven by shared memory interrupts.</p>

## Board Interrupts

If you want to have an interrupt source, the SBS shared memory board can generate PCI interrupts in response to writes to memory partitions of a memory block. You define this behavior through the **WIT** field of shared memory partition structure. Each partition segment has its own **WIT** setting.

To enable a memory location to generate an interrupt, set the **WIT** field as desired, then configure the xPC Target model to generate an interrupt.

The following procedure describes how to configure an entire xPC Target model to accept an interrupt from an SBS shared memory board:

- 1 From the MATLAB Command Window, type

```
getxpcpci
```

This command lists board information for all installed PCI devices that xPC Target knows about.

- 2 If you have multiple SBS shared memory boards, find the board ID for which you want to define the interrupt.
- 3 Find the IRQ specified for your SBS25x0 board.

This is the interrupt source number you need to specify in the **xPC target code generation options** field in step of the following procedure.

- 4 From the MATLAB Command Window, type the name of your Simulink model.

The Simulink model appears.

- 5 From the **Simulation** menu, click **Simulation Parameters**.
- 6 Click the **Real-Time Workshop** tab.
- 7 From the Category list, choose **xPC Target code generation options**.
- 8 Ensure that the **Execution mode** field is set to Real-Time.
- 9 Click the **Real-Time interrupt source** list.

- 10** Select the interrupt number to which the board is set (from step ).
- 11** Click the **I/O board generating the interrupt** list and select the SBS board that is associated with the IRQ. From the list, choose the appropriate board ID:
  - SBS\_25x0\_ID\_0x100
  - SBS\_25x0\_ID\_0x101
  - SBS\_25x0\_ID\_0x102
  - SBS\_25x0\_ID\_0x103
- 12** If your system has multiple boards, at the **PCI slot/ISA base address** parameter, specify the PCI slot that contains the board you are configuring for interrupts.
- 13** Click **OK**.

## **Boards and Blocks – Alphabetical List**

# SBS 2510/2500 Broadcast Memory

---

**Board** SBS 2510/2500 Broadcast Memory

**General Description**

The SBS 2510/2500 Broadcast Memory board is a high-speed fiber optic reflective memory. It can also generate/broadcast interrupts. xPC Target uses this board as part of the shared memory network that you can use to exchange data between computer nodes.

xPC Target supports this board with these driver blocks:

- SBS25x0 init
- SBS25x0 read
- SBS25x0 write

For information about the Change endianness block, see Byte Reversal/Change Endianness.

**Board Characteristics**

Board name	SBS Broadcast Memory
Manufacturer	SBS
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Yes
Multiple board support	Yes

# SBS25x0 init

---

**Purpose** SBS25x0 init block

**Library** xPC Target Library for SBS Technologies

**Block Parameters** Before you begin to configure these block parameters, be sure that you have a predefined node initialization structure. See “Initialize Shared Memory Nodes” on page 31-5.

Each model that uses shared memory must have one 5565 init block for every SBS Broadcast Memory board in the system.

**Node struct**

Enter the name of the predefined node initialization structure.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

`getxpcpci`



**Purpose** SBS25x0 read block

**Library** xPC Target Library for SBS Technologies

**Block Parameters** Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. The 5565 read block requires this structure to specify how Simulink signal values are mapped in the shared memory. It also uses this structure to determine the total size and address of the shared memory. See “Create Shared Memory Partitions” on page 31-3.

**Partition struct**

Enter the name of the predefined shared memory partition structure. The block uses this structure to specify how Simulink signal values map into shared memory.

**Sampletime**

Enter the base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Error Status Port**

Select this check box to monitor the status of the SBS Broadcast Memory board LIER register modes.

# SBS25x0 write

---

**Purpose** SBS25x0 write block

**Library** xPC Target Library for SBS Technologies

**Block Parameters** Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. The 5565 write block requires this structure to specify how Simulink signal values are mapped in the shared memory. It also uses this structure to determine the total size and address of the shared memory. See “Create Shared Memory Partitions” on page 31-3.

**Partition struct**

Enter the name of the predefined shared memory partition and object. The block uses this structure to specify how Simulink signal values map into shared memory.

**Sampletime**

Enter the base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Error Status Port**

Select this check box to monitor the status of the SBS Broadcast Memory board LIER register modes.

# SBS Flex/104A PC/104 IP Carrier Board

---

**Board** SBS Flex/104A PC/104 IP Carrier Board

**General Description** Each IP module must be physically plugged into an IP carrier board such as the Flex/104A. Models containing IP module blocks must also contain corresponding IP carrier blocks. The relationship IP module A is plugged into carrier B' is expressed by selecting the same Carrier ID for both A and B.

The Flex/104A holds up to two IP modules.

xPC Target supports this board with this driver blocks:

- SBS Flex-104A

<b>Board Characteristics</b>	Board name	Flex-104A PC/104 carrier board
	Manufacturer	SBS Technologies
	Bus type	PC/104
	Access method	I/O Mapped
	Multiple block instance support	No
	Multiple board support	Yes

# SBS Flex-104A

---

**Purpose** Flex-104A block

**Library** xPC Target Library for SBS Technologies

**Block** **Carrier ID**

**Parameters** Enter a number to uniquely identify this IP carrier board within the model.

**Base address**

Enter the base address of the board. For example, if the base address is 300 (hexadecimal), enter

0x300

This board permutes the expected pinout of the 50-pin I/O connector for each IP module as follows

Expected Pin	Actual Pin	Expected Pin	Actual Pin	Expected Pin	Actual Pin
1	1	18	35	35	20
2	3	19	37	36	22
3	5	20	39	37	24
4	7	21	41	38	26
5	9	22	43	39	28
6	11	23	45	40	30
7	13	24	47	41	32
8	15	25	49	42	34
9	17	26	2	43	36
10	19	27	4	44	38
11	21	28	6	45	40
12	23	29	8	46	42
13	25	30	10	47	44

<b>Expected Pin</b>	<b>Actual Pin</b>	<b>Expected Pin</b>	<b>Actual Pin</b>	<b>Expected Pin</b>	<b>Actual Pin</b>
14	27	31	12	48	46
15	29	32	14	49	48
16	31	33	16	50	50
17	33	34	18		

# SBS IP-16ADC

---

**Board** SBS IP-16ADC

**General Description** The IP-16ADC I/O board has 16 single or 8 differential analog input (A/D) channels (16-bit) with a combined throughput of 100 kHz.

xPC Target supports this board with this driver block:

- SBS IP-16ADC Analog Input (A/D)

<b>Board Characteristics</b>	Board name	IP-16ADC
	Manufacturer	SBS Technologies
	Bus type	N/A
	Access method	I/O Mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# SBS IP-16ADC Analog Input (A/D)

---

**Purpose** IP-16ADC Analog Input block

**Library** xPC Target Library for SBS Technologies

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

## **Block Parameters**

### **Carrier ID**

Enter the Carrier ID of the IP carrier board into which the IP module is plugged.

In order to use an IP module, it must be physically plugged into an IP carrier board and the model must contain a block representing this carrier board. Each carrier block has a **Carrier ID** parameter, which must be set to a number not shared by any other carrier board block in the model.

### **Carrier slot**

Select the slot on the carrier board into which the IP-16ADC module is plugged. Note that different carrier boards can have different slot capacities.

### **Channel vector**

Enter a vector of numbers between 1 and 16. The channel numbers can occur in any order. For example, to use the first and third analog output (A/D) channels, enter

[1, 3]

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

# SBS IP-16ADC Analog Input (A/D)

---

## Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

Range codes -10 and -5 specify differential channels, which each use two pins on the I/O cable. Range codes 10 and 5 specify single-ended channels, which each use only one pin. Certain combinations of channel numbers and range codes can refer to conflicting physical I/O pins and will cause an error of the form "Bipolar channel x and unipolar channel y use the same I/O pins". Consult the section "I/O Pin Wiring" in the IP-16ADC User's Manual as you select the channel and range vectors to avoid this.

## Sample time

Enter the base sample time or a multiple of the base sample time.



**Board** SBS IP-16DAC

**General Description** The IP-16DAC I/O board has 3 independent precalibrated analog output (D/A) channels (16-bit).

xPC Target supports this board with this driver block:

- SBS IP-16DAC Analog Output (D/A)

<b>Board Characteristics</b>	Board name	IP-16DAC
	Manufacturer	SBS Technologies
	Bus type	N/A
	Access method	I/O Mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# SBS IP-16DAC Analog Output (D/A)

---

**Purpose** IP-16DAC Analog Output block

**Library** xPC Target Library for SBS Technologies

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

## **Block Parameters**

### **Carrier ID**

Enter the Carrier ID of the IP carrier board into which the IP module is plugged.

In order to use an IP module, it must be physically plugged into an IP carrier board and the model must contain a block representing this carrier board. Each carrier block has a **Carrier ID** parameter, which must be set to a number not shared by any other carrier board block in the model.

### **Carrier slot**

Select the slot on the carrier board into which the IP-16DAC module is plugged. Note that different carrier boards can have different slot capacities.

### **Channel vector**

Enter a vector of numbers between 1 and 3. The channel numbers can occur in any order. For example, to use the first and second analog output (D/A) channels, enter

[1,2]

Channel numbers 1, 2, and 3 correspond to DAC A, DAC B, and DAC C respectively.

# SBS IP-16DAC Analog Output (D/A)

---

## Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the jumper settings on the board for DAC A, DAC B, and DAC C.

## Sample time

Base sample time of a multiple of the base sample time.

# SBS IP-DAC

---

**Board** SBS IP-DAC

**General Description** The IP-DAC I/O board has 6 independent precalibrated analog output (D/A) channels (12-bit).

xPC Target supports this board with this driver blocks:

- SBS IP-DAC Analog Output (D/A)

<b>Board Characteristics</b>	Board name	IP-DAC
	Manufacturer	SBS Technologies
	Bus type	N/A
	Access method	I/O Mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# SBS IP-DAC Analog Output (D/A)

**Purpose** IP-DAC Analog Output block

**Library** xPC Target Library for SBS Technologies

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

## **Block Parameters**

### **Carrier ID**

Enter the Carrier ID of the IP carrier board into which the IP module is plugged.

In order to use an IP module, it must be physically plugged into an IP carrier board and the model must contain a block representing this carrier board. Each carrier block has a **Carrier ID** parameter, which must be set to a number not shared by any other carrier board block in the model.

### **Carrier slot**

Select the slot on the carrier board into which the IP-16ADC module is plugged. Note that different carrier boards can have different slot capacities.

### **Channel vector**

Enter a vector of numbers between 1 and 6. The channel numbers can occur in any order. 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.

# SBS IP-DAC Analog Output (D/A)

---

## Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to +5	5
0 to +10	10

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The range settings must correspond to the OUTPUT RANGE SELECTION settings on the board for DAC0 and DAC1 (channel 1 and 2, respectively).

## Sample time

Base sample time of a multiple of the base sample time.

**Board** SBS IP-Digital 24

**General Description** IP-Digital 24 boards have 24 digital I/O lines which can be independently configured for input or output.

xPC Target supports this board with these driver blocks:

- SBS IP-Digital 24 Digital Input
- SBS IP-Digital 24 Digital Output

<b>Board Characteristics</b>	Board name	IP-Digital-24
	Manufacturer	SBS Technologies
	Bus type	N/A
	Access method	I/O Mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# SBS IP-Digital 24 Digital Input

---

**Purpose** IP-Digital 24 Digital Input block

**Library** xPC Target Library for SBS Technologies

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

## **Block Parameters**

### **Carrier ID**

Enter the Carrier ID of the IP carrier board into which the IP module is plugged.

In order to use an IP module, it must be physically plugged into an IP carrier board and the model must contain a block representing this carrier board. Each carrier block has a **Carrier ID** parameter, which must be set to a number not shared by any other carrier board block in the model.

### **Carrier slot**

Select the slot on the carrier board into which the IP-Digital 24 module is plugged. Note that different carrier boards can have different slot capacities.

### **Channel vector**

Enter numbers between 1 and 24 to select the digital input lines to be used. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of lines used.

For example, to use the first three digital inputs, enter

[1,2,3]

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.

# SBS IP-Digital 24 Digital Output

---

**Purpose** IP-Digital 24 Digital Output block

**Library** xPC Target Library for SBS Technologies

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

## **Block Parameters**

### **Carrier ID**

Enter the Carrier ID of the IP carrier board into which the IP module is plugged.

In order to use an IP module, it must be physically plugged into an IP carrier board and the model must contain a block representing this carrier board. Each carrier block has a **Carrier ID** parameter, which must be set to a number not shared by any other carrier board block in the model.

### **Channel vector**

Enter numbers between 1 and 24 to select the digital output lines to be used. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of lines used.

For example, to use the first three digital outputs, enter

[1,2,3]

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.

**Board** SBS IP-HiADC

**General Description** The IP-HiADC I/O board has 16 analog input (A/D) channels (12-bit).  
xPC Target supports this board with this driver block:

- SBS IP-HiADC Analog Input (A/D)

<b>Board Characteristics</b>	Board name	IP-HiADC
	Manufacturer	SBS Technologies
	Bus type	N/A
	Access method	I/O Mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# SBS IP-HiADC Analog Input (A/D)

---

**Purpose** IP-HiADC Analog Input block

**Library** xPC Target Library for SBS Technologies

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

## **Block Parameters**

### **Carrier ID**

Enter the Carrier ID of the IP carrier board into which the IP module is plugged.

In order to use an IP module, it must be physically plugged into an IP carrier board and the model must contain a block representing this carrier board. Each carrier block has a **Carrier ID** parameter, which must be set to a number not shared by any other carrier board block in the model.

### **Carrier slot**

Select the slot on the carrier board into which the IP-HiADC module is plugged. Note that different carrier boards can have different slot capacities.

### **Channel vector**

Enter a vector of numbers between 1 and 16. The channel numbers can occur in any order. For example, to use the first and third analog output (A/D) channels, enter

[1, 3]

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

### **Range**

Select either -5V to 5V or -10V to 10V. This applies to all channels.

# SBS IP-HiADC Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

# SBS IP-Synchro

---

**Board** SBS IP-Synchro

**General Description** IP-Synchro provides two channels of position measurement using synchro, resolver, LVDT, or Inductosyn® transducers.

xPC Target supports this board with this driver block:

- SBS IP-Synchro/Resolver

<b>Board Characteristics</b>	Board name	IP-Synchro
	Manufacturer	SBS Technologies
	Bus type	N/A
	Access method	I/O Mapped
	Multiple block instance support	No
	Multiple board support	Yes

**Purpose** IP-Synchro/Resolver block

**Library** xPC Target Library for SBS Technologies

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Synchro or Resolver	double	angle in radians

## Block Parameters

### Carrier ID

Enter the Carrier ID of the IP carrier board into which the IP-Synchro module is plugged.

In order to use an IP module, it must be physically plugged into an IP carrier board and the model must contain a block representing this carrier board. Each carrier block has a **Carrier ID** parameter, which must be set to a number not shared by any other carrier board block in the model.

### Carrier slot

Select the slot on the carrier board into which the IP-Synchro module is plugged. Note that different carrier boards can have different slot capacities.

### Channel vector

Enter a vector with one or two elements to select the Synchro/Resolver input channels you use with this block. 1 represents channel A and 2 represents channel B. This driver allows the selection of inputs in either order.

For example to use channels A and B in that order, enter

[1,2]

### Precision vector

This must be a scalar or a vector the same length as the channel vector. For each input channel it specifies a precision of either 10,

# SBS IP-Synchro/Resolver

---

12, 14, or 16 bits. If the automatic precision option is in effect, then the selected precision for each channel will be used for the entire run. Otherwise it merely specifies the initial precision.

## **Automatic precision**

Select this check box to have the IP-Synchro automatically change its precision (resolution) in order to track the input velocity.

## **Output format**

This option selects the output status format.

- position — Output position only
- position and status — Output position and status

## **Sample time**

Enter a base sample time or a multiple of the base sample time.



## Board

SBS IP-UD-IHV-16I80

## General Description

The SBS IP-UD-IHV-16I80 board is an IndustryPack board with 16 inputs and 8 outputs. Each IP module must be physically plugged into an IP carrier board such as the PCI-40B. Models containing IP module blocks must also contain corresponding IP carrier blocks. Each carrier block has a unique **Carrier ID** parameter.

xPC Target supports this board with these driver blocks:

- SBS IP-UD-IHV-16I80 Digital Input
- SBS IP-UD-IHV-16I80 Digital Output

The following table illustrates the mapping between the SBS IP-UD-IHV-16I80 board lines and the xPC Target block channels.

---

**Note** In this table, **Byte** and **Bit** correspond to the **Format** parameter 8-bit and 1-bit channel modes of the SBS IP-UD-IHV-16I80 Digital Input and SBS IP-UD-IHV-16I80 Digital Output blocks.

---

# SBS IP-UD-IHV-16I80

SBS Lines	xPC Target Channels		Direction I/O
	Byte	Bit	
1	1	1	Input
2		2	Input
3		3	Input
4		4	Input
5		5	Input
6		6	Input
7		7	Input
8		8	Input
9	2	9	Input
10		10	Input
11		11	Input
12		12	Input
13		13	Input
14		14	Input
15		15	Input
16		16	Input
17	1	1	Output
18		2	Output
19		3	Output
20		4	Output
21		5	Output
22		6	Output
23		7	Output
24		8	Output

## **Board Characteristics**

Board name	SBS IP-UD-IHV-16I80
Manufacturer	SBS Technologies
Bus type	PCI
Multiple block instance support	No
Multiple board support	Yes

# SBS IP-UD-IHV-16I80 Digital Input

---

**Purpose** SBS IP-UD-IHV-16I80 Digital Input block

**Library** xPC Target Library for SBS

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double ( <b>Format:</b> 16 1-bit Channels)	Double: < Threshold = 0.0 > Threshold = 1.0
		uint8 ( <b>Format:</b> Two 8-bit Channels)	uint32: < Threshold, corresponding bit is clear > Threshold, corresponding bit is set

## **Block Parameters**

### **Carrier ID**

Enter a number to uniquely identify this IP carrier board within the model.

### **Carrier slot**

Select the slot on the carrier board into which the module IP-UD-IHV-16I80 is plugged. Note that different carrier boards can have different slot capacities. From the list, select A, B, C, or D.

### **Format**

From the list, select one of the following modes:

- 16 1-bit Channels  
Configures block to accept up to 16 1-bit channels.
- Two 8-bit Channels

Configures block to accept one or two 8-bit channels.

## Channel vector

This value depends on the value of **Format**, as follows:

- If you select the **Format** parameter to be 16 1-bit Channels, enter a vector of numbers between 1 and 16. For example, to use the first and third digital input channels, enter

[1, 3]

- If you select the **Format** parameter to be Two 8-bit Channels, enter a vector of numbers between 1 and 2. For example, to use the first and second digital input channels, enter

[1, 2]

---

**Note** The manufacturer numbers its lines sequentially from 1 to 24, regardless of the line's function. The input lines of this board are numbered 1 to 16. For xPC Target blocks, the channel numbers depend on the setting of the **Format** parameter. If you select the **Format** parameter to be 16 1-bit Channels (bit mode), the channel numbers range from 1 to 16. If you select the **Format** parameter to be Two 8-bit Channels (byte mode), the channel numbers range from 1 to 2. See the mapping table in SBS IP-UD-IHV-16I80.

---

## Threshold

Enter a threshold value. This block reads the input as 1 if the input voltage multiplied by the operational amplifier (op amp) gain is greater than this threshold value. This block reads the input as 0 if the input voltage is less than this threshold value.

The maximum value of **Threshold** is 61.68 volts. It has a granularity of 15 millivolts. See the IP-UD-IHV user documentation for details.

# SBS IP-UD-IHV-16I80 Digital Input

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

# SBS IP-UD-IHV-16I80 Digital Output

**Purpose** SBS IP-UD-IHV-16I80 Digital Output block

**Library** xPC Target Library for SBS Technologies

<b>Scaling of Output to Input</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double ( <b>Format</b> :8 1-bit Channels)	Double: < 0.5, off >0.5, on
		uint8 ( <b>Format</b> :One 8-bit Channel)	uint32: Bit clear, off Bit set, on

## **Block Parameters**

### **Carrier ID**

Enter a number to uniquely identify this IP carrier board within the model.

### **Carrier slot**

#### **Format**

From the list, select one of the following modes:

- 8 1bit Channels  
Configures block to accept up to eight 1-bit channels.
- One 8bit Channels  
Configures block to accept one 8-bit channels.

#### **Channel vector**

This value depends on the value of **Format**, as follows:

- Vector of numbers between 1 and 16

# SBS IP-UD-IHV-16I80 Digital Output

---

If you select the **Format** parameter to be 8 1bit Channels. For example, to use the first and third digital input channels, enter

[1, 3]

- Vector of 1

If you select the **Format** parameter to be One 8bit Channel. For example, to use the first and second digital input channels, enter

[1]

---

**Note** The manufacturer numbers its lines sequentially from 1 to 24, regardless of the line's function. The output lines of this board are numbered 17 to 24. For xPC Target blocks, the channel numbers depend on the setting of the **Format** parameter. If you select the **Format** parameter to be 8 1bit Channels (bit mode), the channel numbers range from 1 to 8. If you select the **Format** parameter to be Two 8-bit Channels (byte mode), the channel number is 1. See the mapping table in SBS IP-UD-IHV-16I80.

---

## 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.

## Final action vector

The final action 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 -1, the block sets the channel to the value specified in the **Final value vector** value for that channel. If you specify a value of 0, the channel remains at the last value attained while the model was running.

## **Final value vector**

The final value vector contains the final value for each output channel. 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 the **Final action vector** is -1, the block sets the channel to this value on model termination.

## **Double buffering**

From the list, select:

- Disable

Disable the use of double-buffering for synchronizing multiple IPs.

- Rising edge

Enable the use of double-buffering for synchronizing multiple IPs on the rising edge.

- Falling edge

Enable the use of double-buffering for synchronizing multiple IPs on the falling edge.

Note that to enable double-buffering, you must have an external clock attached to the board. See the IP-UD-IHV user documentation for details.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

# SBS IP-UD-IHV-8I160

---

## Board

SBS IP-UD-IHV-8I160

## General Description

The SBS IP-UD-IHV-8I160 board is an IndustryPack board with 8 inputs and 16 outputs. Each IP module must be physically plugged into an IP carrier board such as the PCI-40B. Models containing IP module blocks must also contain corresponding IP carrier blocks. Each carrier block has a unique **Carrier ID** parameter.

xPC Target supports this board with these driver blocks:

- SBS IP-UD-IHV-8I160 Digital Input
- SBS IP-UD-IHV-8I160 Digital Output

The following table illustrates the mapping between the SBS IP-UD-IHV-8I160 board lines and the xPC Target block channels.

---

**Note** In this table, **Byte** and **Bit** correspond to the **Format** parameter 8-bit and 1-bit channel modes of the SBS IP-UD-IHV-8I160 Digital Input and SBS IP-UD-IHV-8I160 Digital Output blocks.

---

SBS Lines	xPC Target Channels		Direction I/O
	Byte	Bit	
1	1	1	Input
2		2	Input
3		3	Input
4		4	Input
5		5	Input
6		6	Input
7		7	Input
8		8	Input
9	1	1	Output
10		2	Output
11		3	Output
12		4	Output
13		5	Output
14		6	Output
15		7	Output
16		8	Output
17	2	9	Output
18		10	Output
19		11	Output
20		12	Output
21		13	Output
22		14	Output
23		15	Output
24		16	Output

# SBS IP-UD-IHV-8I160

---

## **Board Characteristics**

Board name	SBS IP-UD-IHV-8I160
Manufacturer	SBS Technologies
Bus type	PCI
Multiple block instance support	No
Multiple board support	Yes

# SBS IP-UD-IHV-8I160 Digital Input

**Purpose** SBS IP-UD-IHV-8I160 Digital Input block

**Library** xPC Target Library for SBS Technologies

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	Volts	Double ( <b>Format:</b> 8 1-bit Channels)	Double: < Threshold = 0.0 > Threshold = 1.0
		uint8 ( <b>Format:</b> One 8-bit Channel)	uint32: < Threshold, corresponding bit is clear > Threshold, corresponding bit is set

## **Block Parameters**

### **Carrier ID**

Enter a number to uniquely identify this IP carrier board within the model.

### **Carrier slot**

Select the slot on the carrier board into which the module IP-UD-IHV-8I160 is plugged. Note that different carrier boards can have different slot capacities. From the list, select A, B, C, or D.

### **Format**

From the list, select one of the following modes:

- 8 1-bit Channels  
Configures block to accept up to eight 1-bit channels.
- One 8-bit Channel

# SBS IP-UD-IHV-8I160 Digital Input

---

Configures block to accept one 8-bit channel.

## Channel vector

This value depends on the value of **Format**, as follows:

- If you select the **Format** parameter to be 8 1bit Channels, enter a vector of numbers between 1 and 8. For example, to use the first and third digital input channels, enter

[1, 3]

- If you select the **Format** parameter to be One 8bit Channels, enter a vector number of 1. For example, to use the single digital input channel, enter

[1]

---

**Note** The manufacturer numbers its lines sequentially from 1 to 24, regardless of the line's function. The input lines of this board are numbered 1 to 8. For xPC Target blocks, the channel numbers depend on the setting of the **Format** parameter. If you select the **Format** parameter to be 8 1bit Channels (bit mode), the channel numbers range from 1 to 8. If you select the **Format** parameter to be One 8-bit Channels (byte mode), the channel number is 1. See the mapping table in SBS IP-UD-IHV-8I160.

---

## Threshold

Enter a threshold value. This block reads the input as 1 if the input voltage multiplied by the operational amplifier (op amp) gain is greater than this threshold value. This block reads the input as 0 if the input voltage is less than this threshold value.

The maximum value of **Threshold** is 61.68 volts. It has a granularity of 15 millivolts. See the IP-UD-IHV user documentation for details.

**Sample time**

Enter a base sample time or a multiple of the base sample time.

# SBS IP-UD-IHV-8I160 Digital Output

---

**Purpose** SBS IP-UD-IHV-8I160 Digital Output block

**Library** xPC Target Library for SBS Technologies

## Scaling Output to Input

### Hardware Input

Volts

### Block Output Data Type

Double (**Format**:16  
1-bit Channels)

### Scaling

Double:  
< 0.5, off  
>0.5, on

uint8 (**Format**:Two  
8-bit Channels)

uint32:  
Bit clear, off  
Bit set, on

## Block Parameters

### Carrier ID

Enter a number to uniquely identify this IP carrier board within the model.

### Carrier slot

### Format

From the list, select one of the following modes:

- 16 1-bit Channels  
Configures block to accept up to 16 1-bit channels.
- Two 8-bit Channels  
Configures block to accept two 8-bit channels.

### Channel vector

This value depends on the value of Format, as follows:

- If you select the **Format** parameter to be 16 1-bit Channels, enter a vector of numbers between 1 and 16. For example, to use the first and third digital input channels, enter



[1, 3]

- If you select the **Format** parameter to be Two 8-bit Channels, enter a vector of numbers between 1 and 2. For example, to use the first and second digital input channels, enter

[1, 2]

---

**Note** The manufacturer numbers its lines sequentially from 1 to 24, regardless of the line's function. The output lines of this board are numbered 9 to 24. For xPC Target blocks, the channel numbers depend on the setting of the **Format** parameter. If you select the **Format** parameter to be 16 1bit Channels (bit mode), the channel numbers range from 1 to 16. If you select the **Format** parameter to be Two 8-bit Channels (byte mode), the channel number is 1 to 2. See the mapping table in SBS IP-UD-IHV-8I160.

---

### **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.

### **Final action vector**

The final action 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 -1, the block sets the channel to the value specified in the **Final value vector** value for that channel. If you specify a value of 0, the channel remains at the last value attained while the model was running.

# SBS IP-UD-IHV-8I160 Digital Output

---

## **Final value vector**

The final value vector contains the final value for each output channel. 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 the **Final action vector** is -1, the block sets the channel to this value on model termination.

## **Double buffering**

From the list, select:

- Disable

Disable the use of double-buffering for synchronizing multiple IPs.

- Rising edge

Enable the use of double-buffering for synchronizing multiple IPs on the rising edge.

- Falling edge

Enable the use of double-buffering for synchronizing multiple IPs on the falling edge.

Note that to enable double-buffering, you must have an external clock attached to the board. See the IP-UD-IHV user documentation for details.

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

**Board** SBS IP-Unidig-E-48

**General Description** IP-Unidig-E-48 boards have 48 digital I/O lines which can be independently configured for input or output.

xPC Target supports this board with these driver blocks:

- SBS IP-Unidig-E-48 Digital Input
- SBS IP-Unidig-E-48 Digital Output

<b>Board Characteristics</b>	Board name	IP-Unidig-E-48
	Manufacturer	SBS Technologies
	Bus type	N/A
	Access method	I/O Mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# SBS IP-Unidig-E-48 Digital Input

---

**Purpose** IP-Unidig-E-48 Digital Input block

**Library** xPC Target Library for SBS Technologies

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

## **Block Parameters**

### **Carrier ID**

Enter the Carrier ID of the IP carrier board into which the IP module is plugged.

In order to use an IP module, it must be physically plugged into an IP carrier board and the model must contain a block representing this carrier board. Each carrier block has a **Carrier ID** parameter, which must be set to a number not shared by any other carrier board block in the model.

### **Carrier slot**

Select the slot on the carrier board into which the IP-Unidig-E-48 module is plugged. Note that different carrier boards can have different slot capacities.

### **Channel vector**

Enter numbers between 1 and 48 to select the digital input lines to be used. This driver allows the selection of individual digital input lines in any order. The number of elements defines the number of lines used.

For example, to use the first three digital inputs, enter

[1,2,3]

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

# SBS IP-Unidig-E-48 Digital Input

---

## **Sample time**

Enter a base sample time or a multiple of the base sample time.

# SBS IP-Unidig-E-48 Digital Output

---

**Purpose** IP-Unidig-E-48 Digital Output block

**Library** xPC Target Library for SBS Technologies

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

## **Block Parameters**

### **Carrier ID**

Enter the Carrier ID of the IP carrier board into which the IP module is plugged.

In order to use an IP module, it must be physically plugged into an IP carrier board and the model must contain a block representing this carrier board. Each carrier block has a **Carrier ID** parameter, which must be set to a number not shared by any other carrier board block in the model.

### **Channel vector**

Enter numbers between 1 and 48 to select the digital output lines to be used. This driver allows the selection of individual digital output lines in any order. The number of elements defines the number of lines used.

For example, to use the first three digital outputs, enter

[1,2,3]

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.

**Board** SBS PCI-40A Carrier Board

**General Description**

Each IP module must be physically plugged into an IP carrier board such as the PCI-40A. Models containing IP module blocks must also contain corresponding IP carrier blocks. The relationship 'IP module A is plugged into carrier B' is expressed by selecting the same Carrier ID for both A and B. The PCI-40A holds up to four IP modules. xPC Target supports this board with this driver block:

- SBS PCI-40A

**Board Characteristics**

Board name	PCI-40A carrier board
Manufacturer	SBS Technologies
Bus type	PCI
Access method	I/O Mapped
Multiple block instance support	No
Multiple board support	Yes

# SBS PCI-40A

---

<b>Purpose</b>	PCI-40A block
<b>Library</b>	xPC Target Library for SBS Technologies
<b>Block Parameters</b>	<b>Carrier ID</b> Enter a number to uniquely identify this IP carrier board within the model. <b>PCI Slot (-1:autosearch)</b> If only one board of this type is in the target PC, enter  -1  to automatically locate the board.  If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type  <code>getxpcpci</code>



**Board** SBS PCI-40B Carrier Board

**General Description**

Each IP module must be physically plugged into an IP carrier board such as the PCI-40B. Models containing IP module blocks must also contain corresponding IP carrier blocks. The relationship “IP module A is plugged into carrier B” is expressed by selecting the same carrier ID for both A and B. The PCI-40B holds up to four IP modules. xPC Target supports this board with this driver block:

- SBS PCI-40B

**Board Characteristics**

Board name	PCI-40B carrier board
Manufacturer	SBS Technologies
Bus type	PCI
Multiple block instance support	No
Multiple board support	Yes

# SBS PCI-40B

---

<b>Purpose</b>	PCI-40B block
<b>Library</b>	xPC Target Library for SBS Technologies
<b>Block Parameters</b>	<b>Carrier ID</b> Enter a number to uniquely identify this IP carrier board within the model. <b>PCI Slot (-1:autosearch)</b> If only one board of this type is in the target PC, enter  -1  to automatically locate the board.  If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type  <code>getxpcpci</code>

# Sensoray

---

This chapter describes the Sensoray I/O boards supported by xPC Target (<http://www.sensoray.com>).

Sensoray 526

Multifunction measurement and control board with eight 16-bit analog inputs, four 16-bit analog outputs, four 24-bit quadrature encoder inputs, and eight digital I/O channels.

Sensoray 626

Multifunction I/O board with 16 14-bit differential analog inputs, four 13-bit analog outputs, three pairs of 24-bit up/down counters (total of six counters), 48 digital I/O channels, watchdog timer, and backup battery support.

“Boards and Blocks — Alphabetical List” on page 32-2

Description of block parameters for Sensoray driver blocks.

## **Boards and Blocks – Alphabetical List**

**Board** Sensoray 526

## General Description

The Sensoray 526 is a multifunction measurement and control board. It has eight 16-bit analog inputs, four 16-bit analog outputs, four 24-bit quadrature encoder inputs, and eight digital I/O channels.

xPC Target supports this board with these driver blocks:

- Sensoray526 AD
- Sensoray526 Dual AD
- Sensoray526 DA
- Sensoray526 Dual DA
- Sensoray526 DI
- Sensoray526 DO
- Sensoray526 Encoder Input

## Board Characteristics

Board name	526
Manufacturer	Sensoray
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Sensoray526 AD

---

**Purpose** Sensoray526 AD block

**Library** xPC Target Library for Sensoray

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts [-10, +10]	Double [-10,+10]	1

## Block Parameters

### Channels

Enter a vector of numbers between 1 and 8. Do not repeat any channel number. For example, to use the first, second and fifth channels, enter

[1, 2, 5]

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

### 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 DIP switch settings on the board. For example, if the base address is 2C0 (hexadecimal), enter

0x2C0

Note that the Sensoray 526 board ships from the manufacturer with a base address of 0x2C0. If you want to change the base address, see the board manufacturer documentation for allowed values.

**Purpose** Sensoray526 Dual AD block

**Library** xPC Target Library for Sensoray

**Note** The Sensoray526 Dual AD block controls two boards simultaneously. This enables simultaneous conversion for both boards. Each board does not have to wait for the other to finish converting data before starting its own conversion. Acquisition overlaps result in considerable time savings.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts [-10,+10]	Double [-10,+10]	1

## Block Parameters

### Board A Channels/Board B Channels

For each board, enter a vector of numbers between 1 and 8. Do not repeat any channel number. For example, to use the first, second and fifth channels, enter

[1, 2, 5]

Note, for optimal performance, equally split the active channels between the two boards. If you need eight channels, use four channels on each board.

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

### Sample time

Base sample time or a multiple of the base sample time.

# Sensoray526 Dual AD

---

## **Board A base address/Board B base address**

Enter the base address of the board. This entry must correspond to the base address DIP switch settings on the board. For example, if the base address is 2C0 (hexadecimal), enter

0x2C0

Note that the Sensoray 526 board ships from the manufacturer with a base address of 0x2C0. If you want to change the base address, see the board manufacturer documentation for allowed values. For dual boards, change the base address of at least one of the boards. The boards cannot have the same base address.



**Purpose** Sensaray526 DA block

**Library** xPC Target Library for Sensaray

<b>Scaling of Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts [10, +10]	Double [10, +10]	1

**Block Parameters**

**Channels**  
Enter a vector of numbers between 1 and 4. Do not repeat any channel number. For example, to use the first, second and fourth channels, enter

```
[1, 2, 4]
```

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

**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. At initialization, the block always uses this value. If **Reset vector** has a value of 1, the board sets the corresponding channel to the value of the **Initial value vector** when execution stops. Enter a double precision voltage value in the range -10 to +10. This must be 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

# Sensoray526 DA

---

set to the initial values between the time the model is downloaded and the time it is started.

## **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 DIP switch settings on the board. For example, if the base address is 2C0 (hexadecimal), enter

0x2C0

Note that the Sensoray 526 board ships from the manufacturer with a base address of 0x2C0. If you want to change the base address, see the board manufacturer documentation for allowed values.

**Purpose** Sensoray526 Dual DA block

**Library** xPC Target Library for Sensoray

**Note** The Sensoray526 Dual AD block controls two boards simultaneously. This enables the block to write the output registers to both boards before the conversion start bits are written on both boards.

Output from the second board changes state approximately 1 to 2 microseconds after the first board. This change is independent of the number of channels in use.

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts [-10,+10]	Double [-10,+10]	1

## Block Parameters

### Board A Channels/Board B Channels

Enter a vector of numbers between 1 and 4. Do not repeat any channel number. For example, to use the first, second and fourth channels, enter

[1, 2, 4]

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

### Board A Reset vector/Board B 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.

## **Board A Initial value vector/Board B Initial value vector**

The initial value vector contains the initial voltage values for the output channels. At initialization, the block always uses this value. If **Reset vector** has a value of 1, the board sets the corresponding channel to the value of the **Initial value vector** when execution stops. Enter a double precision voltage value in the range -10 to +10. This must be 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**

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 DIP switch settings on the board. For example, if the base address is 2C0 (hexadecimal), enter

0x2C0

Note that the Sensoray 526 board ships from the manufacturer with a base address of 0x2C0. If you want to change the base address, see the board manufacturer documentation for allowed values. For dual boards, change the base address of at least one of the boards. The boards cannot have the same base address.

**Purpose** Sensoray526 DI block

**Library** xPC Target Library for Sensoray

**Scaling of Input to Output**

<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL (0 or +5) volts	Double (0 or 1)	1

**Block Parameters**

**Channels**

Enter a vector of numbers between 1 and 8. These are the channels to which the board writes output. Each group, 1 to 4 and 5 to 8, can be either input or output. Do not combine them. For example, if channels 1 to 4 are the input channels, you cannot have output on channel 2. Instead, you can use channels 5 to 8 to be the output channels.

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

**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 DIP switch settings on the board. For example, if the base address is 2C0 (hexadecimal), enter

0x2C0

Note that the Sensoray 526 board ships from the manufacturer with a base address of 0x2C0. If you want to change the base address, see the board manufacturer documentation for allowed values.

# Sensoray526 DO

---

**Purpose** Sensoray526 DO block

**Library** xPC Target Library for Sensoray

## Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL (0 or +5) volts	input < 0.5, output = 0 volts input ≥ 0.5, output = 5 volts	1

## Block Parameters

### Channels

Enter a vector of numbers between 1 and 8. These are the channels to which the board writes output. Each group, 1 to 4 and 5 to 8, can be either input or output. Do not combine them. For example, if channels 1 to 4 are the input channels, you cannot have output on channel 2. Instead, you can use channels 5 to 8 to be the output channels.

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

### 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 logic values for the output channels. At initialization, the block always uses this value. Enter a 0 or 1. If **Reset vector** has a value of 1, the board sets the corresponding channel to the value of the **Initial value vector** when execution stops. 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**

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 DIP switch settings on the board. For example, if the base address is 2C0 (hexadecimal), enter

0x2C0

Note that the Sensoray 526 board ships from the manufacturer with a base address of 0x2C0. If you want to change the base address, see the board manufacturer documentation for allowed values.

# Sensoray526 Encoder Input

---

**Purpose** Sensoray526 Encoder Input block

**Library** xPC Target Library for Sensoray

**Block Parameters** **Channel**

From the list, select 1, 2, 3, or 4. This parameter specifies the encoder input channel that this block reads.

**Index reset mode**

From the list, choose one of the following. This is the event on the index input that resets the counter to the value of **Reset value**.

- Rising edge on index
- Falling edge on index
- Both edges
- None

**Reset value**

The reset value contains the value of the counter when the event that the **Index reset mode** parameter specifies occurs on the index input.

**Initial value**

The initial value vector contains the value that is initially loaded into the counter when model execution starts. The counter is set to the initial value between the time the model is downloaded and the time it is started.

**Output range**

This parameter controls the way the 24-bit integer count is converted to the double precision signal output. From the list, select

- Signed  $\pm 2^{23}$  — If the block interprets the high order bit as a sign bit, the output takes values in the range  $-2^{23}$  to  $(+2^{23})-1$ .
- Unsigned  $0 \rightarrow 2^{24}-1$  — If the block interprets the 24-bit counter as an unsigned quantity, the output is in the range 0 to  $(2^{24})-1$ .



## Count speed

From the list, select a counting mode. Choose one of the following:

- 1x — If counting up, count on the rising edge of CLKA. If counting down, count on the falling edge of CLKA.
- 2x — Count both edges of CLKA, up or down.
- 4x — Count both edges of both CLKA and CLKB, up or down.

Refer to the board manufacturer documentation for details on the counting mode.

## 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 DIP switch settings on the board. For example, if the base address is 2C0 (hexadecimal), enter

0x2C0

Note that the Sensoray 526 board ships from the manufacturer with a base address of 0x2C0. If you want to change the base address, see the board manufacturer documentation for allowed values.

# Sensoray 626

---

## Board

Sensoray 626

## General Description

The Sensoray 626 is a multifunction I/O board. It has 16 14-bit differential analog inputs, four 13-bit analog outputs, three pairs of 24-bit up/down counters (total of six counters), 48 digital I/O channels, watchdog timer, and backup battery support.

You can have up to four Sensoray 626 boards in your system.

xPC Target supports this board with these driver blocks:

- Sensoray 626 Initialize
- Sensoray 626 Analog Input
- Sensoray 626 Analog Output
- Sensoray 626 Digital Input
- Sensoray 626 Digital Output
- Sensoray 626 Encoder
- Sensoray 626 PWM Capture

## Board Characteristics

Board name	626
Manufacturer	Sensoray
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

**Purpose** Sensoray 626 Initialize block

**Library** xPC Target Library for Sensoray

**Note** You must use a Sensoray 626 Initialize block for every physical board that the model uses. One model can have up to four Sensoray 626 Initialize blocks.

**Block Parameters**

**Board**

From the list, select 1, 2, 3, or 4 to represent which of the physical Sensoray 626 boards you want to initialize. Note that this board number is arbitrary. You can reference a physical board as 1, 2, 3, or 4. However, you must be consistent in your board number choice throughout the model.

**Board rev A**

Select this check box if the board you are initializing is a Revision A board. Otherwise, leave this check box unselected.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

Note that the Device ID of this board is 0x7146.

# Sensoray 626 Analog Input

---

**Purpose** Sensoray 626 Analog Input block

**Library** xPC Target Library for Sensoray

**Block Parameters** **Board**

From the list, select 1, 2, 3, or 4 to represent which of the physical Sensoray 626 boards you want to use. Note that you must also add to the model an Initialize block for the same board.

**Channel vector**

Enter a vector of numbers between 1 and 16. This driver allows the selection of individual analog input lines in any order.

The number of elements identifies the number of analog input channels used. For example, to use three analog input channels, enter

[1,2,3]

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

**Range vector**

Enter a range code for each of the channels in the channel vector. The value can be a scalar or a vector that must be the same length as the channel vector. If you specify a scalar value, that setting is used for all channels.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

# Sensoray 626 Analog Input

---

For example, if the first channel is -10 to +10 volts and the second and third channels are -5 to +5 volts, enter

[ -10, -5, -5]

## **Sample time**

Base sample time or a multiple of the base sample time.

# Sensoray 626 Analog Output

---

**Purpose** Sensoray 626 Analog Output block

**Library** xPC Target Library for Sensoray

**Block** **Board**

**Parameters**

From the list, select 1, 2, 3, or 4 to represent which of the physical Sensoray 626 boards you want to use. Note that you must also add to the model an Initialize block for the same board.

**Channel vector**

Enter a vector of numbers between 1 and 4. This driver allows the selection of individual analog output lines in any order. The number of elements identifies the number of analog output channels used. For example, to use three analog output channels, enter

[1, 2, 3]

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

**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 value for each analog output channel when model execution starts. 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. The channel is set to the initial value between the time the model is downloaded and the time it is started.

**Sample time**

Base sample time or a multiple of the base sample time.

# Sensoray 626 Digital Input

---

**Purpose** Sensoray 626 Digital Input block

**Library** xPC Target Library for Sensoray

**Block Parameters**

**Board**  
From the list, select 1, 2, 3, or 4 to represent which of the physical Sensoray 626 boards you want to use. Note that you must also add to the model an Initialize block for the same board.

**Channel vector**  
Enter a vector of numbers between 1 and 48. This driver allows the selection of individual digital input lines in any order. The number of elements identifies the number of digital input channels used. For example, to use three digital input channels, enter

[1, 2, 3]

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

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



<b>Purpose</b>	Sensoray 626 Digital Output block
<b>Library</b>	xPC Target Library for Sensoray
<b>Block Parameters</b>	<p><b>Board</b></p> <p>From the list, select 1, 2, 3, or 4 to represent which of the physical Sensoray 626 boards you want to use. Note that you must also add to the model an Initialize block for the same board.</p> <p><b>Channel vector</b></p> <p>Enter a vector of numbers between 1 and 48. This driver allows the selection of individual digital output lines in any order. The number of elements identifies the number of digital output channels used. For example, to use three digital output channels, enter</p> <p style="text-align: center;">[ 1, 2, 3 ]</p> <p>Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.</p> <p><b>Reset vector</b></p> <p>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.</p> <p><b>Initial value vector</b></p> <p>The initial value vector contains the initial value for each digital output channel when model execution starts. 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. The channel is set to the initial value between the time the model is downloaded and the time it is started.</p>

# Sensoray 626 Digital Output

---

## **Sample time**

Base sample time or a multiple of the base sample time.

**Purpose** Sensoray 626 Encoder block

**Library** xPC Target Library for Sensoray

**Block Parameters**

**Board**

From the list, select 1, 2, 3, or 4 to represent which of the physical Sensoray 626 boards you want to use. Note that you must also add to the model an Initialize block for the same board.

**Channel vector**

Enter a vector of numbers between 1 and 6. This driver allows the selection of individual encoder lines in any order. The number of elements identifies the number of encoder channels used. For example, to use three odd encoder input channels, enter

[1, 3, 5]

Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.

**Count speed vector**

The count speed vector specifies a counting mode for the channel. 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. Choose one of the following.

Code	Quadrature Type
1	1X quadrature
2	2X quadrature
4	4X quadrature

**Index reset mode vector**

This is the event on the index input that resets the channel to the value of the **Reset value vector** value. 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.

# Sensoray 626 Encoder

---

- Rising edge on index
- Falling edge on index
- Both edges
- None

## **Reset value vector**

This parameter contains the preload value for the corresponding channel. This is the value to which the channel is set when the event specified by the **Index reset mode** parameter occurs.

Enter a valid value, as 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.

## **Sample time**

Base sample time or a multiple of the base sample time.

**Purpose** Sensoray 626 PWM Capture block

**Library** xPC Target Library for Sensoray

**Block Parameters** **Board**

From the list, select 1, 2, 3, or 4 to represent which of the physical Sensoray 626 boards you want to use. Note that you must also add to the model an Initialize block for the same board.

**Channel vector**

Enter a vector of numbers between 1 and 3. For example, to use the first two channels, enter

[1, 2]

This parameter refers to one of three counters.

Channel	Function	Pin
1	Encoder(0A)I+	J5-8
2	Encoder(1A)I+	J5-17
3	Encoder(2A)I+	J5-26

**Clock source vector**

This parameter specifies the clock source for the channel. 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. Choose one of the following.

Clock Source Vector	Source
0	Internal 2 MHz clock
1	External clock

**Polarity vector**

This parameter specifies if the channel counts high or low levels. Enter a scalar or a vector that is the same length as the channel

## Sensoray 626 PWM Capture

---

vector. If you specify a scalar value, that setting is used for all channels. Choose one of the following:

<b>Polarity Vector</b>	<b>Source</b>
0	High
1	Low

### **Sample time**

Base sample time or a multiple of the base sample time.

# Softing

---

Due to the complexity of CAN applications, the Softing (<http://www.softing.com>) board drivers that xPC Target supports are not described in this chapter. Instead, see Chapter 4, “CAN I/O Support” and Chapter 5, “CAN I/O Support for FIFO” for details.





# Systran

---

This chapter describes the Systran reflective shared memory boards supported by xPC Target (<http://www.systran.com>).

Before You Start (p. 34-2)

Guidelines for setting up structures for shared memory.

Systran Shared Memory Structure Reference (p. 34-6)

Shared memory reference for structures

Boards and Blocks — Alphabetical List (p. 34-16)

Description of block parameters for Systran driver blocks.

## Before You Start

xPC Target uses a model for reflective memory (also known as shared memory) that includes Simulink blocks, for the shared memory drivers, and MATLAB structures for defining shared memory and node initialization partitions. The topics in this section are

- “Create Shared Memory Partitions” on page 34-2 — Before you begin to use SCRAMNet+ SC150 shared memory blocks, define a partition structure that defines how you want to allocate shared memory.
- “Initialize Shared Memory Nodes” on page 34-4 — Before you begin to use SCRAMNet+ SC150 shared memory blocks, define a node initialization structure that defines how shared memory is allocated (partitioned) and how the board is configured.

### Create Shared Memory Partitions

To use the xPC Target SCRAMNet+ SC150 shared memory blocks to read, write, pack, or unpack data, you must define a partition structure. SCRAMNet+ SC150 shared memory drivers use MATLAB structures to define shared memory partitions. A partition structure describes how you want to allocate (or partition) the shared memory. xPC Target allocates shared memory with bundles of data that are packed into memory partitions. The following SCRAMNet+ SC150 blocks use shared memory partition structures:

- Systran Scramnet SC150 read and Systran Scramnet SC150 write
- Shared Memory Pack and Shared Memory Unpack

After you define the shared memory partitions, you can add SCRAMNet+ SC150 shared memory driver blocks to your Simulink model. Create a shared memory structure in one of the following ways. See “Shared Memory Partition Structure” on page 34-6 for the complete list of fields for the structure.

The following description refers to the `completepartitionstruct` command.  
Type

```
help completepartitionstruct
```

for a description of the command.

- Using the `completepartitionstruct` command at the MATLAB Command Window, create a default partition structure. For example, type

```
completepartitionstruct([], 'scramnet')

ans =
    Address: '0x0'
      Type: 'uint32'
       Size: '1'
  Alignment: '4'
        RIE: 'off'
        TIE: 'off'
  ExtTrigger1: 'off'
  ExtTrigger2: 'off'
        HIPRO: 'off'
    Internal: [1x1 struct]
```

- At the MATLAB Command Window, create a user-defined partition structure. The easiest way to do this is to create an M-file, partially define a structure, load that M-file into the MATLAB workspace, and supplement the resulting structure with a call to the `completepartitionstruct` function. For example

```
Partition(1).Address='0x5000';
Partition(1).Type='int8';
Partition(1).Size='10';
Partition(2).Type='uint16';
Partition(2).Size='5';
Partition(3).Type='double';
Partition(3).Size='3';
Partition(4).Type='uint8';
Partition(4).Size='[2, 3]';
```

This example defines a partition with four segments.

- The `Address` field is optional. Only specify this field for the first segment of a partition. The elements of a partition are defined as a continuous memory block from the first address. The following segments extrapolate their addresses from the first segment. If you have fragmented memory, use multiple partitions and SCRAMNet+ SC150 read/write blocks to work with the memory.

- The Type and Size fields are required for all fields in the partition structure.
- The base addresses of subsequent segments are fully defined by the data type, size, and alignment of the preceding segment.

You can then call the `completpartitionstruct()` command to full populate the partition structure.

## Initialize Shared Memory Nodes

To use the xPC Target SCRAMNet+ SC150 shared memory, you must define a node initialization structure. SCRAMNet+ SC150 shared memory drivers use MATLAB structures to define shared memory node initializations. A node initialization structure describes the shared memory partition (see “Create Shared Memory Partitions” on page 34-2) and the SCRAMNet+ SC150 board configuration, including interrupt configurations. The following SCRAMNet+ SC150 block requires shared memory node initialization structures:

- SC150 init

After you define the node initialization partition, you can add SCRAMNet+ SC150 shared memory driver blocks to your Simulink model. Create a shared memory structure in one of the following ways. See “Shared Memory Node Initialization Structure” on page 34-10 for the complete list of fields for the structure.

The following description refers to the `completenodestruct` command. Type

```
help completenodestruct
```

for a description of the command.

- Using the `completenodestruct` command at the MATLAB Command Window, create a default node initialization structure. For example, type

```
node=completenodeconstruct([], 'scramnet')
```

```
node =
```

```
    Interface: [1x1 struct]  
    Partitions: [1x1 struct]
```

- A user-defined node structure, create with m-code or from the MATLAB Command Window and supplement the resulting structure with a call to the `completenodeconstruct` function.

## Systran Shared Memory Structure Reference

You do not need to use all the fields of a partition or node initialization structure. However, knowing the possible structure fields will be helpful when you are creating any of the structures.

This section includes the following topics:

- “Shared Memory Partition Structure” on page 34-6 — Description of the partition node structure fields
- “Shared Memory Node Initialization Structure” on page 34-10 — Description of the node initialization structure fields

### Shared Memory Partition Structure

A shared memory partition structure has the following fields:

```

Address: '0x0'
Type: 'uint32'
Size: '1'
Alignment: '4'
RIE: 'off'
TIE: 'off'
ExtTrigger1: 'off'
ExtTrigger2: 'off'
HIPRO: 'off'
Internal: [1x1 struct]

```

where

Partition Fields	Description
<b>Address</b>	Specifies the base address (in hexadecimal) of the memory partition within the node’s shared memory space. The default value is '0x0', the first location in shared memory. Note that the base address is byte aligned.

<b>Partition Fields</b>	<b>Description</b>
<b>Type</b>	<p>Specifies the data type of the memory segment. Specify one of the following types:</p> <ul style="list-style-type: none"> <li>• double</li> <li>• float</li> <li>• uint8</li> <li>• int8</li> <li>• uint16</li> <li>• int16</li> <li>• uint32</li> <li>• int32</li> <li>• boolean (a single byte represents a boolean value)</li> </ul> <p>The default value is 'uint32'. A minimum partition size is 32 bits.</p>
<b>Size</b>	<p>Specifies the dimension and size of the memory segment. You can enter a scalar value or a value with the [m,n] format. The default value is '1'.</p> <ul style="list-style-type: none"> <li>• scalar — Treats the Size entry as the specification of the length of a non-oriented array or vector</li> <li>• [m,n] — Treats the Size entry as an array dimension. The total number of elements in this segment is m*n.</li> </ul>
<b>Alignment</b>	<p>If another partition precedes this partition, defines the byte alignment of this segment. Specify one of the following alignment values: 1, 2, 3, or 4. The default value is '4'. This value forces a double word boundary alignment for all elements.</p>

<b>Partition Fields</b>	<b>Description</b>
<b>RIE</b>	<p>Specifies whether or not this partition can receive interrupts (Receive Interrupt Register (RIE)). Specify 'off', 'first', 'all', 'last'. The default value is 'off'.</p> <p>'off' — Prevents the partition from receiving interrupts</p> <p>'first' — Allows only the first double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p> <p>'all' — Allows all memory locations of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p> <p>'last' — Allows only the last double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p>
<b>TIE</b>	<p>Specifies whether or not this partition can transmit interrupts (Transmit Enable (TIE)). Specify 'off', 'first', 'all', 'last'. The default value is 'off'.</p> <p>'off' — Prevents the partition from transmitting interrupts</p> <p>'first' — Allows only the first double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p> <p>'all' — Allows all memory locations of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p> <p>'last' — Allows only the last double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p>



<b>Partition Fields</b>	<b>Description</b>
<b>ExtTrigger1</b>	<p>If this partition receives a write access, specifies whether or not this partition can generate a trigger signal to an external connector. Specify 'off', 'first', 'all', 'last'. The default value is 'off'.</p> <p>'off' — Prevents the partition from generating signals</p> <p>'first' — Allows only the first double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p> <p>'all' — Allows all memory locations of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p> <p>'last' — Allows only the last double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p>
<b>ExtTrigger2</b>	<p>If this partition receives a write access, specifies whether or not this partition can generate a trigger signal to an external connector. Specify 'off', 'first', 'all', 'last'. The default value is 'off'.</p> <p>'off' — Prevents the partition from generating signals</p> <p>'first' — Allows only the first double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p> <p>'all' — Allows all memory locations of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p> <p>'last' — Allows only the last double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.</p>

Partition Fields	Description
<b>HIPRO</b>	Specifies whether or not the elements in this partition can be transmitted as one network message. Specify 'off', or 'on'. The default value is 'off'.  'off' — Prevents the partition from transmitting the elements as one message  'on' — Allows the partition to transmit the elements as one message
<b>Internal</b>	Reserved for internal use.

## Shared Memory Node Initialization Structure

A node initialization structure has the following fields:

Interface: [1x1 struct]  
Partitions: [1x1 struct]

where

<b>Node Structure Fields</b>	<b>Description</b>
<b>Interface</b>	<p>Specifies settings for the board Control/Status Register (CSR). The Interface structure has the following fields. Refer to the SCRAMNet+ SC150 product documentation for a description of the CSR and its operation modes.</p> <ul style="list-style-type: none"> <li>• <b>Mode</b> — Configures board modes (see “Board Mode” on page 34-11)</li> <li>• <b>Timeout</b> — Enables the board to set the timeout value (see “Board Timeout” on page 34-13)</li> <li>• <b>DataFilter</b> — Controls the data filtering operation (see “Board Data Filter” on page 34-13)</li> <li>• <b>VirtualPaging</b> — Controls the board virtual paging operation (see “Virtual Paging” on page 34-14)</li> <li>• <b>Interrupts</b> — Enables the board to generate and receive interrupts from the network (see “Board Interrupts” on page 34-14)</li> <li>• <b>Internal</b> — Reserved for internal use</li> </ul>
<b>Partitions</b>	Stores the shared memory segments (see “Create Shared Memory Partitions” on page 34-2)

### Board Mode

The SCRAMNet+ SC150 board has a number of modes that you can set through the `Interface.Mode` field. The `Interface Mode` fields set the corresponding bits in the CSR. To display the board mode fields, type

```
>> node.Interface.Mode
ans =
    NetworkCommunicationsMode: 'TransmitReceive'
           InsertNode: 'on'
    DisableFiberOpticLoopback: 'on'
           EnableWireLoopback: 'off'
    DisableHostToMemoryWrite: 'off'
```

```

WriteOwnSlotEnable: 'off'
MessageLengthLimit: '256'
VariableLengthMessagesOnNetwork: 'off'
HIPROEnable: 'off'
MultipleMessages: 'on'
NoNetworkErrorCorrection: 'on'
MechanicalSwitchOverride: 'on'
DisableHoldoff: 'on'

```

These modes have the following values:

Field	Values	Default	CSR
<b>NetworkCommunications Mode</b>	'none', 'receiveonly', 'transmitonly', 'transmit receive'	'transmit receive'	CSR3[8..15]
<b>InsertNode</b>	'off', 'on'	'on'	CSR0[0..1]
<b>DisableFiberOptic Loopback</b>	'off', 'on'	'on'	CSR2[6]
<b>EnableWire Loopback</b>	'off', 'on'	'off'	CSR2[7]
<b>DisableHost ToMemory Write</b>	'off', 'on'	'off'	CSR2[8]
<b>WriteOwnSlotEnable</b>	'off', 'on'	'off'	CSR2[9]
<b>Message LengthLimit</b>	'256', '1024'	'256'	CSR2[11]
<b>Variable Length MessagesOn Network</b>	'off', 'on'	'off'	CSR2[12]
<b>HIPROEnable</b>	'off', 'on'	'off'	CSR2[13]
<b>Multiple Messages</b>	'off', 'on'	'on'	CSR2[14]
<b>NoNetwork Error Correction</b>	'off', 'on'	'on'	CSR2[15]
<b>Mechanical Switch Override</b>	'off', 'on'	'on'	CSR8[11]
<b>Disable Holdoff</b>	'off', 'on'	'on'	CSR8[11]

## Board Timeout

The SCRAMNet+ SC150 board allows you to set the network timeout through the `Interface.Timeout` field. The `Interface.Timeout` fields set the corresponding bits in the CSR.

To display the timeout fields, type

```
>> node.Interface.Timeout
ans =
      NumOfNodesInRing: '2'
      TotalCableLengthInM: '2'
```

These fields have the following values:

Field	Values	Default	CSR
<b>NumOfNodes InRing</b>	'0' .. '255'	'2'	CSR5
<b>TotableCable LengthInM</b>	'0' .. 'n'	'2'	CSR5

Refer to the SCRAMNet+ SC150 product documentation for a description of these fields.

## Board Data Filter

The SCRAMNet+ SC150 board allows you to set the data filter operation through the `Interface.DataFilter` field. The `Interface.DataFilter` fields set the corresponding bits in the CSR.

```
>> node.Interface.DataFilter
ans =
      EnableTransmitDataFilter: 'off'
      EnableLower4KBytesForDataFilter: 'off'
>>
```

These fields have the following values:

Field	Values	Default	CSR
<b>Enable TransmitData Filter</b>	'off', 'on'	'off'	CSR0[10]
<b>EnableLower4KBytesFor DataFilter</b>	'off', 'on'	'off'	CSR0[11]

### Virtual Paging

The SCRAMNet+ SC150 board allows you to set the bits of the Virtual Paging Register operation through the `Interface.VirtualPaging` field. The `Interface.VirtualPaging` fields set the corresponding bits in the CSR.

```
>> node.Interface.VirtualPaging
ans =
  VirtualPagingEnable: 'off'
  VirtualPageNumber: '0'
```

These fields have the following values:

Field	Values	Default	CSR
<b>VirtualPagingEnable</b>	'off', 'on'	'off'	CSR12[0]
<b>VirtualPage Number</b>	'0'..'2047'	'0'	CSR12[5..15]

### Board Interrupts

The SCRAMNet+ SC150 board allows you to specify the interrupt sources transmitted and received between the nodes of the network. You can set these bits through the `Interface.Interrupts` field. The `Interface.Interrupts` fields set the corresponding bits in the CSR.

```
>> node.Interface.Interrupts
ans =
      HostInterrupt: 'off'
InterruptOnMemoryMaskMatch: 'off'
      OverrideReceiveInterrupt: 'off'
      InterruptOnError: 'off'
      NetworkInterrupt: 'off'
OverrideTransmitInterrupt: 'off'
      InterruptOnOwnSlot: 'off'
      ReceiveInterruptOverride: 'off'
```

These fields have the following values:

Field	Values	Default	CSR
<b>HostInterrupt</b>	'off', 'on'	'off'	CSR0[3]
<b>InterruptOn MemoryMask Match</b>	'off', 'on'	'off'	CSR0[5]
<b>Override Receive Interrupt</b>	'off', 'on'	'off'	CSR0[6]
<b>InterruptOn Error</b>	'off', 'on'	'off'	CSR0[7]
<b>Network Interrupt</b>	'off', 'on'	'off'	CSR0[8]
<b>Override Transmit Interrupt</b>	'off', 'on'	'off'	CSR0[9]
<b>InterruptOn OwnSlot</b>	'off', 'on'	'off'	CSR2[10]
<b>Receive Interrupt Override</b>	'off', 'on'	'off'	CSR8[10]

## **Boards and Blocks – Alphabetical List**



**Board** Systran SCRAMNet+ SC150 PCI

## General Description

The SCRAMNet+ SC150 PCI board is a real-time reflective PCI memory board. It can also generate/broadcast interrupts. xPC Target uses this board as part of the shared memory network that you can use to exchange data between computer nodes.

xPC Target supports this board with these driver blocks:

- Systran Scramnet SC150 init
- Systran Scramnet SC150 read
- Systran Scramnet SC150 write

For information about the Change endianness block, see [Byte Reversal/Change Endianness](#).

## Board Characteristics

Board name	SCRAMNet+ SC150
Manufacturer	Systran
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Systran Scramnet SC150 init

---

**Purpose** Systran Scramnet SC150 init block

**Library** xPC Target Library for Systran

**Note** Before you begin to configure these block parameters, be sure that you have a predefined node initialization structure. See “Initialize Shared Memory Nodes” on page 34-4.

**Block Parameters** Each model that uses shared memory must have one SC150 init block for every SCRAMNet+ SC150 board in the system.

**Node struct**  
Enter the name of the predefined node initialization structure.

**PCI Slot (-1:autosearch)**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** Systran Scramnet SC150 read block

**Library** xPC Target Library for Systran

**Note** Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. The SC150 read block requires this structure to specify how Simulink signal values are mapped in the shared memory. It also uses this structure to determine the total size and address of the shared memory. See “Create Shared Memory Partitions” on page 34-2.

**Block Parameters**

**Partition struct**

Enter the name of the predefined shared memory partition structure. The block uses this structure to specify how Simulink signal values map into shared memory.

**Error port**

Select this check box to monitor the status of the Systran Scramnet board CSR1 register modes. The enabled port is of type `uint16`.

**Sampletime**

Enter the base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Systran Scramnet SC150 write

---

**Purpose** Systran Scramnet SC150 write block

**Library** xPC Target Library for Systran

**Note** Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. The SC150 write block requires this structure to specify how Simulink signal values are mapped in the shared memory. It also uses this structure to determine the total size and address of the shared memory. See “Create Shared Memory Partitions” on page 34-2.

## **Block Parameters**

### **Partition struct**

Enter the name of the predefined shared memory partition structure. The block uses this structure to specify how Simulink signal values map into shared memory.

### **Error port**

Select this check box to monitor the status of the Systran Scramnet board CSR1 register modes. The enabled port is of type uint16.

### **Sampletime**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# United Electronic Industries (UEI)

---

This chapter describes the UEI groups of boards supported by xPC target (<http://www.ueidaq.com>).

“Grouping the UEI Boards” on page 35-5	Explanation of the grouping of the UEI boards.
“Analog Input Frame Driver Blocks” on page 35-7	Explanation of analog input frame driver blocks.
UEI PD2-MF 12-Bit Series	16 or 64 single or 8 or 32 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 1.25 MHz. The gain varies from 1 to 8 or 1 to 1000. The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PD2-MF 14-Bit Series	16 or 64 single or 8 or 32 differential analog input (A/D) channels (14-bit). The gain varies from 1 to 8 or 1 to 1000. The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input lines, and 16 digital output lines.

UEI PD2-MF 16-Bit Series	16 or 64 single or 8 or 32 differential analog input (A/D) channels (16-bit). The gain varies from 1 to 8 or 1 to 1000. The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PD2-MFS 12-Bit Series	4 or 8 single analog input (A/D) channels (12-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PD2-MFS 14-Bit Series	4 or 8 single analog input (A/D) channels (12-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PD2-MFS 16-Bit Series	4 or 8 single analog input (A/D) channels (16-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PDXI-MF 12-Bit Series	16 or 64 single or 8 or 32 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 1.25 MHz. The gain varies from 1 to 8 or 1 to 1000. The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PDXI-MF 14-Bit Series	16 or 64 single or 8 or 32 differential analog input (A/D) channels (14-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.

UEI PDXI-MF 16-Bit Series	16 or 64 single or 8 or 32 differential analog input (A/D) channels (16-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PDXI-MFS 12-Bit Series	4 or 8 single analog input (A/D) channels (12-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PDXI-MFS 14-Bit Series	4 or 8 single analog input (A/D) channels (14-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PDXI-MFS 16-Bit Series	4 or 8 single analog input (A/D) channels (16-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines.
UEI PD2-AO Series	8, 16, or 32 analog output (D/A) channels (16-bit) with a maximum sample rate of 100 kHz per channel. They also have 8 digital input and 8 digital output lines.
UEI PD2-DIO-64	64 digital I/O channels in 16-bit ports, three counters/timers, two enhanced synchronous serial interface (ESSI) ports, and four high-speed (100 ns) IRQ lines.
UEI PD2-DIO-128	128 digital I/O channels in 16-bit ports, three counters/timers, two enhanced synchronous serial interface (ESSI) ports, and four high-speed (100 ns) IRQ lines.

UEI PD2-DIO-128i	64 optoisolated digital input and 64 optoisolated digital output ports in 16-bit ports.
UEI PDL-DIO-64	Lab boards with 64 digital I/O channels in 16-bit ports, three counters/timers, two enhanced synchronous serial interface (ESSI) ports, and four high-speed (100 ns) IRQ lines.
UEI PDXI-DIO-64	64 digital I/O channels in 16-bit ports, three counters/timers, two enhanced synchronous serial interface (ESSI) ports, and four high-speed (100 ns) IRQ lines.
UEI PDXI-AO Series	8, 16, or 32 analog output (D/A) channels (16 bit) with a maximum sampling rate of 100 kHz per channel. They also have 8 digital input lines, and 8 digital output lines.
“Boards and Blocks — Alphabetical List” on page 35-14	Description of block parameters for UEI driver blocks.



## Grouping the UEI Boards

The United Electronic Industries (UEI) PowerDAQ board series contains a large number of boards. The board names follow a standard pattern:

[Form Factor]-[Board Type]-[Channel]-[Speed]/[Resolution][Gain]

For example, one UEI board is named PD2-MF-16-1M/12L. The possibilities for the parts of the name are

- **Form Factor** — The form factor can be one of two things. PD2 denotes a 32-bit, 33 MHz PCI bus board. PDXI indicates a 32-bit, 33 MHz PXI/PCI bus board.
- **Board Type** — The board type can be one of three things. MF indicates a multifunction board. MFS indicates a multifunction board with sample and hold. AO indicates an analog output board.
- **Channel** — For MF and MFS board types, the channel is the number of analog input channels. For AO board types, the channel is the number of analog output channels.
- **Speed** — The speed is the number of samples per second supported by the board.
- **Resolution** — For MF and MFS boards, the Resolution is the analog input resolution. For AO board types, the Resolution is the analog output resolution.
- **Gain** — Gain is denoted by letters that represent ranges. DG indicates a gain of 1, 2, 5, or 10. L indicates a range of 1, 10, 100, or 1000. H indicates a range of 1, 2, 4, or 8.

For example, the PD2-MF-16-1M/12L is a PCI multifunction board with sixteen 12-bit analog input channels supporting one million (1M) samples per second with available gains of 1, 10, 100, or 1000.

### Changing the Board Associated with the Block

Note that the board displayed on the block is the current board type. To specify a different board, double-click on the block. In the **Board type** list, choose a different board.

## **Getting Information on a Specific Board**

The boards in this manual are grouped by Form Factor, Board Type, and Resolution. For more information on a specific board, open the MATLAB Help browser. Click on the **Search** tab. Verify that the **Product Filter** is set to **All** or that xPC Target is one of your selected products. In the **Search type** list, choose **Full Text**. In the **Search for** text box, type the name of your board. Click **Go**. Browse through the search results for more information on your board.

## Analog Input Frame Driver Blocks

The following UEI PowerDAQ board series have Analog Input frame driver blocks:

- PD2-MF — 12, 14, and 16-bit series
- PD2-MFS — 12, 14, and 16-bit series
- PDXI-MF — 12, 14, and 16-bit series
- PDXI-MFS — 12, 14, and 16-bit series

In frame-based mode, the boards for these driver blocks send interrupts to xPC Target. The xPC Target model is executed when each frame completes on the board. For proper use of the UEI frame driver blocks, note the IRQ values for each UEI board in the model and configure those boards appropriately with the model Configuration Parameters dialog. The topics in this section describe

- “Notes on Master and Slave Boards” on page 35-7 — Note the listed master and slave boards usage notes.
- “Interrupt Numbers” on page 35-8 — Look for and note the correct UEI board IRQ number. You use the number that the target machine BIOS assigns.
- “Interrupt Configuration” on page 35-10 — Configure the model with the correct UEI board IRQ number using the model simulation parameters dialog.
- “Example Models” on page 35-12 — Refer to the example UEI frame model examples in the xpcdemos directory.

### Notes on Master and Slave Boards

Before you begin, observe the following usage notes for the UEI boards in a master/slave configuration:

- You should set the slowest board as the master. If you do not want to use the slowest board as the master, you can explicitly set the **Acquisition frequency** parameter of the master board driver block to one that enables the slowest board in the configuration to work. You can derive the maximum speed of the board from the board name.

- Slave boards do not need to be identical to the master board.
- In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.
- The number of channels in use on the master does not need to be the same as those used on any of the slave boards.
- Connect the master and its slave(s) with the J6 connectors on the boards and the appropriate cables. This ensures that the acquisition clock and the scan clock from the master board is sent to all slave boards.
- The number of slaves you can have is limited by the number of connection points available on the cable and the availability of PCI slots.

## Interrupt Numbers

This topic describes how you determine the interrupt number to which the board is set. The methods vary depending on your hardware configuration.

This topic includes

- “Single UEI Board” on page 35-9 — How to use `getxpcpci` to get information about the PCI devices installed on the target machine. This method works if you have one UEI board in your system. The target machine BIOS assigns the IRQ number.
- “Guidelines for Multiple UEI Boards” on page 35-9 — How to get information about the PCI devices if you have two or more UEI boards configured in a master/slave configuration. This is an iterative method that requires you to reboot the target machine with the insertion of each UEI board.

---

**Note** To ensure that the PCI BIOS sets up the plugged-in PCI cards properly, disable (set to No) the Plug-and-Play (PnP) operating system feature. The xPC Target kernel is not a PnP operating system; you must ensure that this feature is disabled or PCI devices will not work on xPC Target.

---

## Single UEI Board

If you have one UEI board for your target machine, that board is the master.

- 1 Insert the UEI board in a slot of your target machine.
- 2 Reboot the target machine.
- 3 From the MATLAB Command Window, type

```
getxpcpci all
```

A list of the installed PCI devices and their relevant information is displayed. Included in this information is the IRQ. Note the IRQ number for the UEI board for which you are configuring the frame driver block.

- 4 Compare this board's IRQ number with the Ethernet Controller IRQ number, also listed in the `getxpcpci` display.

These numbers should not be the same. If they are, move the UEI board to another slot and reboot the target machine.

- 5 Assuming a unique IRQ UEI board number, note the IRQ, bus, and slot number for the UEI board.

Once you know your UEI board and its IRQ number, go to “Interrupt Configuration” on page 35-10 for further configuration details.

## Guidelines for Multiple UEI Boards

If you have two or more UEI boards in a master/slave configuration, you should try the following procedure. Note which UEI board you want to use as the master.

- 1 Insert a board into the target machine.
- 2 Reboot the target machine.
- 3 From a MATLAB Command Window running on the host machine, type

```
getxpcpci all
```

- 4 Note the bus and slot information for the board you just installed.

- 5** Repeat Steps 1 to 4 for each UEI board you want in your configuration.
- 6** Note the IRQ for the UEI board you designate as the master after you install all the boards.
- 7** Check this board's IRQ number with the Ethernet Controller IRQ number, also listed in the getxpcpci display.

These numbers should not be the same. If they are, move the master UEI board to another slot, reboot the target machine, and check the master IRQ number again. Ignore the slave board IRQ numbers.

- 8** Assuming a unique IRQ UEI board number, note the IRQ, bus, and slot number for the UEI board.

Once you know the bus and slot information for the UEI boards, and the master's IRQ number, go to "Interrupt Configuration" on page 35-10 for further configuration details.

## **Interrupt Configuration**

After you add a frame block to a model, use the following procedures to set the simulation parameters to use the interrupt from the UEI board rather than the timer interrupt when running the model. This section assumes that you know the appropriate IRQ numbers for the boards you are configuring. If you do not, see "Interrupt Numbers" on page 35-8.

This topic includes

- "Simulation Parameters" on page 35-10 — If the UEI A/D frame block is not in a Function-Call Subsystem called from an IRQ Source block, use this procedure.
- "IRQ Source Block" on page 35-11 — If the UEI A/D frame block is in a Function-Call Subsystem called from an IRQ Source block, use this procedure.

## **Simulation Parameters**

If the UEI A/D frame block is not in a Function-Call Subsystem called from an IRQ Source block,

- 1** From the model, select **Simulation > Configuration parameters**.
- 2** Select the **Real-Time Workshop** node.
- 3** In the **Target selection** section, from the **System target file** list, browse to and select `xpctarget.tlc`.
- 4** In the **xPC Target options** node, from the **Real-time interrupt source** list, select the IRQ number you have jumpered on the board.

---

**Note** This number must not match the IRQ number for the Ethernet controller.

---

- 5** In the same node, from the **I/O board generating the interrupt** list, select the value `UEI-MFx`. This specifies that the UEI MFx board generates the interrupt.
- 6** In the same node, for the **PCI slot/ISA base address** parameter, enter the same PCI address as for the UEI MFx Frame block **PCI slot** parameter.
  - Only enter -1 (for autodetection) if this is the only UEI board in the target system.
  - If more than one UEI board is or will be in the target system, enter the bus and slot number using the format `[bus, slot]`.
- 7** Click **OK** and save the model.

### **IRQ Source Block**

If the UEI A/D frame block is in a Function-Call Subsystem called from an IRQ Source block,

- 1** In the model, select the IRQ Source block.
- 2** From the **IRQ line number** list, select the IRQ number.
- 3** From the **I/O board generating the interrupt** list, select the value `UEI-MFx`. This specifies that a UEI MFx board generates the interrupt.

---

**Note** This number must not match the IRQ number for the Ethernet controller.

---

- 4** For the **PCI slot** parameter, enter the same PCI address as for the UEI MFx Frame block **PCI slot** parameter.
  - Only enter -1 (for autodetection) if this is the only UEI board in the target system.
  - If more than one UEI board is or will be in the target system, enter the bus and slot number using the format [bus, slot].
- 5** Click **OK**.
- 6** Repeat this procedure for each IRQ Source block calling separate subsystems.

## Example Models

Example models that illustrate the use of UEI frame driver blocks are in the `xpcdemos` directory. Correct operation of these models requires that the interrupt source be set correctly in the simulation parameters dialog.

Note that these examples use the UEI PD2-MF-16-333/16H driver block. Replace the block as appropriate for your target system.

- `xpcUEIFrame.mdl` — A simple frame-based model that outputs to an xPC Target scope. The model runs each time the board completes a frame of data.
- `xpcUEIMasterSlaveframe.mdl` — A master/slave frame-based model.
- `xpcUEIasync.mdl` — A model that implements a UEI A/D frame block in a Function-Call Subsystem called by an IRQ Source block. To set the interrupt for the UEI board, you work with the IRQ Source block parameters.

In this example, the interval at which the IRQ Source block triggers the Function-Call Subsystem is the board frame completion time. The rest of the model runs at the model sample time. This latter point implies that there may or may not be a new frame of output data in the output signal



when the rest of the model executes. However, if you place all processing of the data in the Function-Call Subsystem, this is not an issue.

---

**Note** Even if the frame time at which the board runs is set to the same time as the model sample time, because the times derive from different sources they will drift relative to each other.

---

- `xpcUEIDualasync.mdl` — A model that implements multiple UEI A/D frame blocks in Function-Call Subsystems called by IRQ Source blocks. To set the interrupt for the UEI boards, you work with the IRQ Source block parameters.
- `xpcFrameLoop.mdl` — A model that illustrates how to perform sample-based processing on a frame of data. Because the fastest interrupt available is the frame completion interrupt, no part of the model can execute at a faster rate than frame completion. The For Iterator block executes once per frame completion and loops through the data.

## **Boards and Blocks – Alphabetical List**

## Board Series

UEI PD2-MF 12-Bit Series

## General Board Series Description

The PD2-MF 12-bit series contains I/O boards with 16 or 64 single or 8 or 32 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 1.25 MHz or 3 MHz. The gain varies from 1 to 8 or 1 to 1000. The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-MF 12-Bit Series Analog Input (A/D)
- UEI PD2-MF 12-Bit Series Frame Analog Input
- UEI PD2-MF 12-Bit Series Analog Output (D/A)
- UEI PD2-MF 12-Bit Series Digital Input
- UEI PD2-MF 12-Bit Series Digital Output

## Board Characteristics

Board type	PD2-MF-16-1M/12L PD2-MF-16-1M/12H PD2-MF-64-1M/12L PD2-MF-64-1M/12H PD2-MF-16-3M/12L PD2-MF-16-3M/12H PD2-MF-64-3M/12L PD2-MF-64-3M/12H
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PD2-MF 12-Bit Series Analog Input (A/D)

---

**Purpose** PD2-MF 12-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

**Gain vector**  
Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

**Mux settling time factor**  
This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

**Range**  
Select the voltage range from the list provided. This applies to all channels.

# UEI PD2-MF 12-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 12-Bit Series Frame Analog Input

---

**Purpose** PD2-MF 12-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[1 5]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

**Gain vector**  
Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

**Mux settling time vector (slow bit)**  
Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PD2-MF 12-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts

# UEI PD2-MF 12-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## **Input coupling mode**

From the list, select one from the following list of input coupling modes:

- Single Ended
- Differential

## **Frame size**

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 512. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 256.

## **Output format**

From the list, select either Frame or Vector.

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

## **Scan clock source**

Select Internal or External to identify the clock source for the frame scans.

## **Scan time**

Enter the time (in seconds) between scans as the interval.



# UEI PD2-MF 12-Bit Series Frame Analog Input

---

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time** value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## Block is in an ISR

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## Slave board

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

# UEI PD2-MF 12-Bit Series Frame Analog Input

---

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## UEI PD2-MF 12-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 12-Bit Series Analog Output (D/A)

---

**Purpose** PD2-MF 12-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the output channels 1 or 2. For example, to use the first and second analog output (D/A) channels, enter

[1 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0.

**Reset vector**  
The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

**Initial value vector**  
The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output channels. The channels are set to these values between the time the model is downloaded and the time it is started.

# UEI PD2-MF 12-Bit Series Analog Output (D/A)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 12-Bit Series Digital Input

---

**Purpose** PD2-MF 12-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

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

**PCI Slot (-1:autosearch)**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# UEI PD2-MF 12-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 12-Bit Series Digital Output

---

**Purpose** PD2-MF 12-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

**Reset vector**  
The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

**Initial value vector**  
The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output



# UEI PD2-MF 12-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 14-Bit Series

---

## Board Series

UEI PD2-MF 14-Bit Series

## General Board Series Description

The PD2-MF 14-bit series contains I/O boards with 16 or 64 single or 8 or 32 differential analog input (A/D) channels (14-bit). The gain varies from 1 to 8 or 1 to 1000. The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input lines, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-MF 14-Bit Series Analog Input (A/D)
- UEI PD2-MF 14-Bit Series Frame Analog Input
- UEI PD2-MF 14-Bit Series Analog Output (D/A)
- UEI PD2-MF 14-Bit Series Digital Input
- UEI PD2-MF 14-Bit Series Digital Output

## Board Characteristics

Board type	PD2-MF-16-400/14L PD2-MF-16-400/14H PD2-MF-64-400/14L PD2-MF-64-400/14H PD2-MF-16-2M/14H PD2-MF-64-2M/14H
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI, ISA
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PD2-MF 14-Bit Series Analog Input (A/D)

**Purpose** PD2-MF 14-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

### **Mux settling time factor**

This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

### **Range**

Select the voltage range from the list provided. This applies to all channels.

# UEI PD2-MF 14-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 14-Bit Series Frame Analog Input

**Purpose** PD2-MF 14-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[ 1 5 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

### **Mux settling time vector (slow bit)**

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PD2-MF 14-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts

# UEI PD2-MF 14-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## **Input coupling mode**

From the list, select one from the following list of input coupling modes:

- Single Ended
- Differential

Refer to the UEI PowerDAQ documentation for input connections.

## **Frame size**

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 512. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 256.

## **Output format**

From the list, select either Frame or Vector.

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

## **Scan clock source**

Select Internal or External to identify the clock source for the frame scans.

# UEI PD2-MF 14-Bit Series Frame Analog Input

---

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time** value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\text{Frametime} = \text{Framesize} \times \text{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## Block is in an ISR

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## Slave board

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)



# UEI PD2-MF 14-Bit Series Frame Analog Input

---

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## UEI PD2-MF 14-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 14-Bit Series Analog Output (D/A)

**Purpose** PD2-MF 14-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels 1 or 2. For example, to use the first and second analog output (D/A) channels, enter

[ 1 2 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output channels. The channels are set to these values between the time the model is downloaded and the time it is started.

# UEI PD2-MF 14-Bit Series Analog Output (D/A)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 14-Bit Series Digital Input

**Purpose** PD2-MF 14-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

## UEI PD2-MF 14-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 14-Bit Series Digital Output

**Purpose** PD2-MF 14-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[ 1 3 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PD2-MF 14-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



## Board Series

UEI PD2-MF 16-Bit Series

## General Board Series Description

The PD2-MF 16-bit series contains I/O boards with 16 or 64 single or 8 or 32 differential analog input (A/D) channels (16-bit). The gain varies from 1 to 8 or 1 to 1000. The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-MF 16-Bit Series Analog Input (A/D)
- UEI PD2-MF 16-Bit Series Frame Analog Input
- UEI PD2-MF 16-Bit Series Analog Output (D/A)
- UEI PD2-MF 16-Bit Series Digital Input
- UEI PD2-MF 16-Bit Series Digital Output

## Board Characteristics

Board type	PD2-MF-16-50/16H PD2-MF-16-333/16L PD2-MF-16-333/16H PD2-MF-64-333/16L PD2-MF-64-333/16H PD2-MF-16-500/16L PD2-MF-16-500/16H PD2-MF-64-500/16L PD2-MF-64-500/16H
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI, ISA
Access method	Memory mapped

# UEI PD2-MF 16-Bit Series

---

Multiple block instance support	No
Multiple board support	Yes

# UEI PD2-MF 16-Bit Series Analog Input (A/D)

**Purpose** PD2-MF 16-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

### **Mux settling time factor**

This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

### **Range**

Select the voltage range from the list provided. This applies to all channels.

## UEI PD2-MF 16-Bit Series Analog Input (A/D)

---

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 16-Bit Series Frame Analog Input

**Purpose** PD2-MF 16-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[ 1 5 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

### **Mux settling time vector (slow bit)**

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PD2-MF 16-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts

# UEI PD2-MF 16-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## **Input coupling mode**

From the list, select one from the following list of input coupling modes:

- Single Ended
- Differential

Refer to the UEI PowerDAQ documentation for input connections.

## **Frame size**

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 512. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 256.

## **Output format**

From the list, select either Frame or Vector.

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

## **Scan clock source**

Select Internal or External to identify the clock source for the frame scans.

# UEI PD2-MF 16-Bit Series Frame Analog Input

---

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time** value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## Block is in an ISR

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## Slave board

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the



# UEI PD2-MF 16-Bit Series Frame Analog Input

---

box unchecked for the master board. (Note that a single board configuration is considered a master board.)

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

# UEI PD2-MF 16-Bit Series Frame Analog Input

---

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 16-Bit Series Analog Output (D/A)

**Purpose** PD2-MF 16-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels 1 or 2. For example, to use the first and second analog output (D/A) channels, enter

[1 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output channels. The channels are set to these values between the time the model is downloaded and the time it is started.

# UEI PD2-MF 16-Bit Series Analog Output (D/A)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 16-Bit Series Digital Input

**Purpose** PD2-MF 16-Bit Series Digital Input block

**Library** xPC Target Library for UEI

## Scaling Input to Output

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

## Block Parameters

### Board type

Select the specific board type from the list provided.

### Channel vector

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

## UEI PD2-MF 16-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MF 16-Bit Series Digital Output

**Purpose** PD2-MF 16-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[ 1 3 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PD2-MF 16-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



## Board Series

UEI PD2-MFS 12-Bit Series

## General Board Series Description

The PD2-MFS 12-bit series contains I/O boards with 4 or 8 single analog input (A/D) channels (12-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-MFS 12-Bit Series Analog Input (A/D)
- UEI PD2-MFS 12-Bit Series Frame Analog Input
- UEI PD2-MFS 12-Bit Series Analog Output (D/A)
- UEI PD2-MFS 12-Bit Series Digital Input
- UEI PD2-MFS 12-Bit Series Digital Output

## Board Characteristics

Board type	PD2-MFS-4-1M/12 PD2-MFS-4-1M/12DG PD2-MFS-8-1M/12 PD2-MFS-8-1M/12DG
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI, ISA
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PD2-MFS 12-Bit Series Analog Input (A/D)

---

**Purpose** PD2-MFS 12-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

### **Mux settling time factor**

This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

### **Range**

Select the voltage range from the list provided. This applies to all channels.

# UEI PD2-MFS 12-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 12-Bit Series Frame Analog Input

---

**Purpose** PD2-MFS 12-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[1 5]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type:

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

### **Mux settling time vector (slow bit)**

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PD2-MFS 12-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block samptime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts

# UEI PD2-MFS 12-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## Frame size

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 1024. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 512.

## Output format

From the list, select either Frame or Vector.

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

## Scan clock source

Select Internal or External to identify the clock source for the frame scans.

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time**

# UEI PD2-MFS 12-Bit Series Frame Analog Input

---

value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## **Block is in an ISR**

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## **Slave board**

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

# UEI PD2-MFS 12-Bit Series Frame Analog Input

---

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.



# UEI PD2-MFS 12-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 12-Bit Series Analog Output (D/A)

---

**Purpose** PD2-MFS 12-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels 1 or 2. For example, to use the first and second analog output (D/A) channels, enter

[1 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output channels. The channels are set to these values between the time the model is downloaded and the time it is started.

# UEI PD2-MFS 12-Bit Series Analog Output (D/A)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 12-Bit Series Digital Input

---

**Purpose** PD2-MFS 12-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

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

**PCI Slot (-1:autosearch)**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# UEI PD2-MFS 12-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 12-Bit Series Digital Output

---

**Purpose** PD2-MFS 12-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

**Reset vector**  
The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

**Initial value vector**  
The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PD2-MFS 12-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 14-Bit Series

---

## Board Series

UEI PD2-MFS 14-Bit Series

## General Board Series Description

The PD2-MFS 14-bit series contains I/O boards with 4 or 8 single analog input (A/D) channels (14-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-MFS 14-Bit Series Analog Input (A/D)
- UEI PD2-MFS 14-Bit Series Frame Analog Input
- UEI PD2-MFS 14-Bit Series Analog Output (D/A)
- UEI PD2-MFS 14-Bit Series Digital Input
- UEI PD2-MFS 14-Bit Series Digital Output

## Board Characteristics

Board type	PD2-MFS-4-500/14 PD2-MFS-4-500/14DG PD2-MFS-8-500/14 PD2-MFS-8-500/14DG PD2-MFS-4-800/14 PD2-MFS-4-800/14DG PD2-MFS-8-800/14 PD2-MFS-8-800/14DG PD2-MFS-4-2M/14 PD2-MFS-4-2M/14DG PD2-MFS-8-2M/14 PD2-MFS-8-2M/14DG
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI, ISA
Access method	Memory mapped



Multiple block instance support	No
Multiple board support	Yes

# UEI PD2-MFS 14-Bit Series Analog Input (A/D)

---

**Purpose** PD2-MFS 14-Bit Series Analog Input series

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

### **Mux settling time factor**

This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

### **Range**

Select the voltage range from the list provided. This applies to all channels.

# UEI PD2-MFS 14-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 14-Bit Series Frame Analog Input

**Purpose** PD2-MFS 14-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[1 5]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

### **Mux settling time vector (slow bit)**

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PD2-MFS 14-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds. For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts
- 0-10 Volts

# UEI PD2-MFS 14-Bit Series Frame Analog Input

---

- 0-5 Volts

## Frame size

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 1024. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 512.

## Output format

From the list, select either Frame or Vector.

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

## Scan clock source

Select Internal or External to identify the clock source for the frame scans.

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time** value and reiterate the process. Always rebuild the model when you change this value.

# UEI PD2-MFS 14-Bit Series Frame Analog Input

---

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## **Block is in an ISR**

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## **Slave board**

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

# UEI PD2-MFS 14-Bit Series Frame Analog Input

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# UEI PD2-MFS 14-Bit Series Analog Output (D/A)

**Purpose** PD2-MFS 14-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels 1 or 2. For example, to use the first and second analog output (D/A) channels, enter

[1 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output channels. The channels are set to these values between the time the model is downloaded and the time it is started.

# UEI PD2-MFS 14-Bit Series Analog Output (D/A)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 14-Bit Series Digital Input

**Purpose** PD2-MFS 14-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

## UEI PD2-MFS 14-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 14-Bit Series Digital Output

**Purpose** PD2-MFS 14-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[ 1 3 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PD2-MFS 14-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board Series

UEI PD2-MFS 16-Bit Series

## General Board Series Description

The PD2-MFS 16-bit series contains I/O boards with 4 or 8 single analog input (A/D) channels (16-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-MFS 16-Bit Series Analog Input (A/D)
- UEI PD2-MFS 16-Bit Series Frame Analog Input
- UEI PD2-MFS 16-Bit Series Analog Output (D/A)
- UEI PD2-MFS 16-Bit Series Digital Input
- UEI PD2-MFS 16-Bit Series Digital Output

## Board Characteristics

Board type	PD2-MFS-4-300/16 PD2-MFS-4-300/16DG PD2-MFS-8-300/16 PD2-MFS-8-300/16DG PD2-MFS-4-500/16 PD2-MFS-4-500/16DG PD2-MFS-8-500/16 PD2-MFS-8-500/16DG
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI, ISA
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PD2-MFS 16-Bit Series Analog Input (A/D)

---

**Purpose** PD2-MFS 16-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

### **Mux settling time factor**

This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

### **Range**

Select the voltage range from the list provided. This applies to all channels.



# UEI PD2-MFS 16-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 16-Bit Series Frame Analog Input

---

**Purpose** PD2-MFS 16-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[1 5]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

### **Mux settling time vector (slow bit)**

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PD2-MFS 16-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts

# UEI PD2-MFS 16-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## Frame size

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 1024. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 512.

## Output format

From the list, select either Frame or Vector.

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

## Scan clock source

Select Internal or External to identify the clock source for the frame scans.

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time**

# UEI PD2-MFS 16-Bit Series Frame Analog Input

---

value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## **Block is in an ISR**

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## **Slave board**

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

# UEI PD2-MFS 16-Bit Series Frame Analog Input

---

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# UEI PD2-MFS 16-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 16-Bit Series Analog Output (D/A)

---

**Purpose** PD2-MFS 16-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels 1 or 2. For example, to use the first and second analog output (D/A) channels, enter

[1 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output channels. The channels are set to these values between the time the model is downloaded and the time it is started.



# UEI PD2-MFS 16-Bit Series Analog Output (D/A)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 16-Bit Series Digital Input

---

**Purpose** PD2-MFS 16-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

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

**PCI Slot (-1:autosearch)**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# UEI PD2-MFS 16-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-MFS 16-Bit Series Digital Output

---

**Purpose** PD2-MFS 16-Bit Series Digital Output

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

**Reset vector**  
The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

**Initial value vector**  
The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PD2-MFS 16-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 12-Bit Series

---

## Board Series

UEI PDXI-MF 12-Bit Series

## General Board Series Description

The PDXI-MF 12-bit series contains I/O boards with 16 or 64 single or 8 or 32 differential analog input (A/D) channels (12-bit) with a maximum sample rate of 1.25 MHz or 3 MHz. The gain varies from 1 to 8 or 1 to 1000. The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PDXI-MF 12-Bit Series Analog Input (A/D)
- UEI PDXI-MF 12-Bit Series Frame Analog Input
- UEI PDXI-MF 12-Bit Series Analog Output (D/A)
- UEI PDXI-MF 12-Bit Series Digital Input
- UEI PDXI-MF 12-Bit Series Digital Output

## Board Characteristics

Board type	PDXI-MF-16-1M/12L PDXI-MF-16-1M/12H PDXI-MF-64-1M/12L PDXI-MF-64-1M/12H PDXI-MF-16-3M/12L PDXI-MF-16-3M/12H PDXI-MF-64-3M/12L PDXI-MF-64-3M/12H
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI
Access method	Memory mapped

Multiple block instance support	No
Multiple board support	Yes

# UEI PDXI-MF 12-Bit Series Analog Input (A/D)

---

**Purpose** PDXI-MF 12-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

**Gain vector**  
Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

**Mux settling time factor**  
This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

**Range**  
Select the voltage range from the list provided. This applies to all channels.



# UEI PDXI-MF 12-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 12-Bit Series Frame Analog Input

---

**Purpose** PDXI-MF 12-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[1 5]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

**Gain vector**  
Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

**Mux settling time vector (slow bit)**  
Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PDXI-MF 12-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts

# UEI PDXI-MF 12-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## **Input coupling mode**

From the list, select one from the following list of input coupling modes:

- Single Ended
- Differential

Refer to the UEI PowerDAQ documentation for input connections.

## **Frame size**

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 1024. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 512.

## **Output format**

From the list, select either Frame or Vector.

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

## **Scan clock source**

Select Internal or External to identify the clock source for the frame scans.

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time** value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## Block is in an ISR

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## Slave board

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

# UEI PDXI-MF 12-Bit Series Frame Analog Input

---

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## UEI PDXI-MF 12-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 12-Bit Series Analog Output (D/A)

---

**Purpose** PDXI-MF 12-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels between 1 and 16. For example, to use the first and second analog output (D/A) channels, enter

[1 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for PD2-AO and PDXI-AO series boards is 8, 16, or 32 depending on the specific board type. For all other boards series the maximum channel number is 2.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output



# UEI PDXI-MF 12-Bit Series Analog Output (D/A)

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 12-Bit Series Digital Input

---

**Purpose** PDXI-MF 12-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

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

**PCI Slot (-1:autosearch)**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# UEI PDXI-MF 12-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 12-Bit Series Digital Output

---

**Purpose** PDXI-MF 12-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

**Reset vector**  
The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

**Initial value vector**  
The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PDXI-MF 12-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 14-Bit Series

---

## Board Series

UEI PDXI-MF 14-Bit Series

## General Board Series Description

The PDXI-MF 14-bit series contains I/O boards with 16 or 64 single or 8 or 32 differential analog input (A/D) channels (14-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PDXI-MF 14-Bit Series Analog Input (A/D)
- UEI PDXI-MF 14-Bit Series Frame Analog Input
- UEI PDXI-MF 14-Bit Series Analog Output (D/A)
- UEI PDXI-MF 14-Bit Series Digital Input
- UEI PDXI-MF 14-Bit Series Digital Output

## Board Characteristics

Board type	PDXI-MF-16-400/14L PDXI-MF-16-400/14H PDXI-MF-64-400/14L PDXI-MF-64-400/14H PDXI-MF-16-2M/14H PDXI-MF-64-2M/14H
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PDXI-MF 14-Bit Series Analog Input (A/D)

**Purpose** PDXI-MF 14-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

### **Mux settling time factor**

This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

### **Range**

Select the voltage range from the list provided. This applies to all channels.

# UEI PDXI-MF 14-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# UEI PDXI-MF 14-Bit Series Frame Analog Input

**Purpose** PDXI-MF 14-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[1 5]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

### **Mux settling time vector (slow bit)**

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PDXI-MF 14-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts

# UEI PDXI-MF 14-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## **Input coupling mode**

From the list, select one from the following list of input coupling modes:

- Single Ended
- Differential

Refer to the UEI PowerDAQ documentation for input connections.

## **Frame size**

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 1024. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 512.

## **Output format**

From the list, select either Frame or Vector.

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

## **Scan clock source**

Select Internal or External to identify the clock source for the frame scans.

# UEI PDXI-MF 14-Bit Series Frame Analog Input

---

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time** value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\text{Frametime} = \text{Framesize} \times \text{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## Block is in an ISR

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## Slave board

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## UEI PDXI-MF 14-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 14-Bit Series Analog Output (D/A)

**Purpose** PDXI-MF 14-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels between 1 and 16. For example, to use the first and second analog output (D/A) channels, enter

[ 1 2 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for PD2-AO and PDXI-AO series boards is 8, 16, or 32 depending on the specific board type. For all other boards series the maximum channel number is 2.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output

# UEI PDXI-MF 14-Bit Series Analog Output (D/A)

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# UEI PDXI-MF 14-Bit Series Digital Input

**Purpose** PDXI-MF 14-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## Block Parameters

### Board type

Select the specific board type from the list provided.

### Channel vector

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

## UEI PDXI-MF 14-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 14-Bit Series Digital Output

**Purpose** PDXI-MF 14-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[ 1 3 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PDXI-MF 14-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board Series

UEI PDXI-MF 16-Bit Series

## General Board Series Description

The PDXI-MF 16-bit series contains I/O boards with 16 or 64 single or 8 or 32 differential analog input (A/D) channels (16-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PDXI-MF 16-Bit Series Analog Input (A/D)
- UEI PDXI-MF 16-Bit Series Frame Analog Input
- UEI PDXI-MF 16-Bit Series Analog Output (D/A)
- UEI PDXI-MF 16-Bit Series Digital Input
- UEI PDXI-MF 16-Bit Series Digital Output

## Board Characteristics

Board type	PDXI-MF-16-333/16L PDXI-MF-16-333/16H PDXI-MF-64-333/16L PDXI-MF-64-333/16H PDXI-MF-16-500/16L PDXI-MF-16-500/16H PDXI-MF-64-500/16L PDXI-MF-64-500/16H
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PDXI-MF 16-Bit Series Analog Input (A/D)

---

**Purpose** PDXI-MF 16-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

### **Mux settling time factor**

This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

### **Range**

Select the voltage range from the list provided. This applies to all channels.

# UEI PDXI-MF 16-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 16-Bit Series Frame Analog Input

---

**Purpose** PDXI-MF 16-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[1 5]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

**Gain vector**  
Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

**Mux settling time vector (slow bit)**  
Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding



# UEI PDXI-MF 16-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts

# UEI PDXI-MF 16-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## **Input coupling mode**

From the list, select one from the following list of input coupling modes:

- Single Ended
- Differential

Refer to the UEI PowerDAQ documentation for input connections.

## **Frame size**

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 1024. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 512.

## **Output format**

From the list, select either Frame or Vector.

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

## **Scan clock source**

Select Internal or External to identify the clock source for the frame scans.

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time** value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## Block is in an ISR

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## Slave board

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

# UEI PDXI-MF 16-Bit Series Frame Analog Input

---

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## UEI PDXI-MF 16-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 16-Bit Series Analog Output (D/A)

---

**Purpose** PDXI-MF 16-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels between 1 and 16. For example, to use the first and second analog output (D/A) channels, enter

[1 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for PD2-AO and PDXI-AO series boards is 8, 16, or 32 depending on the specific board type. For all other boards series the maximum channel number is 2.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output

# UEI PDXI-MF 16-Bit Series Analog Output (D/A)

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 16-Bit Series Digital Input

---

**Purpose** PDXI-MF 16-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

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

**PCI Slot (-1:autosearch)**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with



# UEI PDXI-MF 16-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MF 16-Bit Series Digital Output

---

**Purpose** PDXI-MF 16-Bit Series Digital Output

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

**Reset vector**  
The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

**Initial value vector**  
The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PDXI-MF 16-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 12-Bit Series

---

## Board Series

UEI PDXI-MFS 12-Bit Series

## General Description

The PDXI-MFS 12-bit series contains I/O boards with 4 or 8 single analog input (A/D) channels (12-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PDXI-MFS 12-Bit Series Analog Input (A/D)
- UEI PDXI-MFS 12-Bit Series Frame Analog Input
- UEI PDXI-MFS 12-Bit Series Analog Output (D/A)
- UEI PDXI-MFS 12-Bit Series Digital Input
- UEI PDXI-MFS 12-Bit Series Digital Output

## Board Characteristics

Board type	PDXI-MFS-4-1M/12 PDXI-MFS-4-1M/12DG PDXI-MFS-8-1M/12 PDXI-MFS-8-1M/12DG
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PDXI-MFS 12-Bit Series Analog Input (A/D)

**Purpose** PDXI-MFS 12-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

### **Mux settling time factor**

This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

### **Range**

Select the voltage range from the list provided. This applies to all channels.

# UEI PDXI-MFS 12-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 12-Bit Series Frame Analog Input

**Purpose** PDXI-MFS 12-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[ 1 5 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

### **Mux settling time vector (slow bit)**

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PDXI-MFS 12-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts



# UEI PDXI-MFS 12-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## Frame size

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 1024. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 512.

## Output format

From the list, select either Frame or Vector.

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

## Scan clock source

Select Internal or External to identify the clock source for the frame scans.

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time**

# UEI PDXI-MFS 12-Bit Series Frame Analog Input

---

value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## **Block is in an ISR**

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## **Slave board**

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

# UEI PDXI-MFS 12-Bit Series Frame Analog Input

---

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# UEI PDXI-MFS 12-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 12-Bit Series Analog Output (D/A)

**Purpose** PDXI-MFS 12-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels between 1 and 16. For example, to use the first and second analog output (D/A) channels, enter

[ 1 2 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for PD2-AO and PDXI-AO series boards is 8, 16, or 32 depending on the specific board type. For all other boards series the maximum channel number is 2.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output

# UEI PDXI-MFS 12-Bit Series Analog Output (D/A)

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 12-Bit Series Digital Input

**Purpose** PDXI-MFS 12-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## Block Parameters

### Board type

Select the specific board type from the list provided.

### Channel vector

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# UEI PDXI-MFS 12-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# UEI PDXI-MFS 12-Bit Series Digital Output

**Purpose** PDXI-MFS 12-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[ 1 3 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PDXI-MFS 12-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board Series

UEI PDXI-MFS 14-Bit Series

## General Board Series Description

The PDXI-MFS 14-bit series contains I/O boards with 4 or 8 single analog input (A/D) channels (14-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PDXI-MFS 14-Bit Series Analog Input (A/D)
- UEI PDXI-MFS 14-Bit Series Frame Analog Input
- UEI PDXI-MFS 14-Bit Series Analog Output (D/A)
- UEI PDXI-MFS 14-Bit Series Digital Input
- UEI PDXI-MFS 14-Bit Series Digital Output

## Board Characteristics

Board type	PDXI-MFS-4-500/14 PDXI-MFS-4-500/14DG PDXI-MFS-8-500/14 PDXI-MFS-8-500/14DG PDXI-MFS-4-800/14 PDXI-MFS-4-800/14DG PDXI-MFS-8-800/14 PDXI-MFS-8-800/14DG PDXI-MFS-4-2M/14 PDXI-MFS-4-2M/14DG PDXI-MFS-4-2M/14H PDXI-MFS-8-2M/14 PDXI-MFS-8-2M/14DG
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI

## UEI PDXI-MFS 14-Bit Series

---

Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PDXI-MFS 14-Bit Series Analog Input (A/D)

**Purpose** PDXI-MFS 14-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

### **Mux settling time factor**

This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

### **Range**

Select the voltage range from the list provided. This applies to all channels.

# UEI PDXI-MFS 14-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 14-Bit Series Frame Analog Input

**Purpose** PDXI-MFS 14-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[ 1 5 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

### **Mux settling time vector (slow bit)**

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PDXI-MFS 14-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block sampletime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts



# UEI PDXI-MFS 14-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## Frame size

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 1024. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 512.

## Output format

From the list, select either Frame or Vector.

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

## Scan clock source

Select Internal or External to identify the clock source for the frame scans.

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time**

# UEI PDXI-MFS 14-Bit Series Frame Analog Input

---

value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## **Block is in an ISR**

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## **Slave board**

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

# UEI PDXI-MFS 14-Bit Series Frame Analog Input

---

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

## UEI PDXI-MFS 14-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 14-Bit Series Analog Output (D/A)

**Purpose** PDXI-MFS 14-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels between 1 and 16. For example, to use the first and second analog output (D/A) channels, enter

[ 1 2 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for PD2-AO and PDXI-AO series boards is 8, 16, or 32 depending on the specific board type. For all other boards series the maximum channel number is 2.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output

# UEI PDXI-MFS 14-Bit Series Analog Output (D/A)

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 14-Bit Series Digital Input

**Purpose** PDXI-MFS 14-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## Block Parameters

### Board type

Select the specific board type from the list provided.

### Channel vector

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# UEI PDXI-MFS 14-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# UEI PDXI-MFS 14-Bit Series Digital Output

**Purpose** PDXI-MFS 14-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[ 1 3 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PDXI-MFS 14-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board Series

UEI PDXI-MFS 16-Bit Series

## General Board Series Description

The PDXI-MFS 16-bit series contains I/O boards with 4 or 8 single analog input (A/D) channels (16-bit). The boards also have 2 analog output (D/A) channels (12-bit), 16 digital input, and 16 digital output lines. xPC Target does not support the counter/timers on these boards.

xPC Target supports this series of boards with these driver blocks:

- UEI PDXI-MFS 16-Bit Series Analog Input (A/D)
- UEI PDXI-MFS 16-Bit Series Frame Analog Input
- UEI PDXI-MFS 16-Bit Series Analog Output (D/A)
- UEI PDXI-MFS 16-Bit Series Digital Input
- UEI PDXI-MFS 16-Bit Series Digital Output

## Board Characteristics

Board type	PDXI-MFS-4-300/16 PDXI-MFS-4-300/16DG PDXI-MFS-8-300/16 PDXI-MFS-8-300/16DG PDXI-MFS-4-500/16 PDXI-MFS-4-500/16DG PDXI-MFS-8-500/16 PDXI-MFS-8-500/16DG
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PDXI-MFS 16-Bit Series Analog Input (A/D)

---

**Purpose** PDXI-MFS 16-Bit Series Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, use the first and third analog input (A/D) channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

**Gain vector**  
Enter a vector of numbers to select the gains for each of the channels. Available gains vary depending on board type. MF PGL gains may be set to 1, 10, 100, or 1000. MF PGH gains may be set to 1, 2, 4, or 8. MF DG-option gains may be set to 1, 2, 5, or 10.

**Mux settling time factor**  
This vector must be the same length as the channel vector. It contains values of 0 or 1. If 1, the corresponding channel has a longer settling time. This is useful when using a high gain such as 100 or 1000.

**Range**  
Select the voltage range from the list provided. This applies to all channels.

# UEI PDXI-MFS 16-Bit Series Analog Input (A/D)

---

## **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 16-Bit Series Frame Analog Input

---

**Purpose** PDXI-MFS 16-Bit Series Frame Analog Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers that make up one scan. For example, to read the first and fifth channels, enter

[1 5]

The channel numbers can occur in any order. Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 4, 8, 16, or 64 depending on the board type.

### **Gain vector**

Enter a vector of numbers to specify the gains for each of the channels in the scan. Enter a single value to use the same gain for all vectors in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.
- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

### **Mux settling time vector (slow bit)**

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding

# UEI PDXI-MFS 16-Bit Series Frame Analog Input

---

channel has a longer settling time. To get an accurate reading, you might want to specify a 1 for channels that have higher gains, such as 100 or 1000. Refer to the UEI PowerDAQ documentation for further information on the longer settling time. This time depends on the board type.

---

**Caution** With frame based acquisition, the additional time needed for the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that xPC Target does not detect. For example, on a PD2-MF-xx-333/16H board, each nonslow acquisition takes 3 microseconds. If the slow bit is set to 1, that same acquisition takes more than 10 microseconds. With xPC Target, the scan is performed at the maximum rate for a given board. This means that the slow acquisition takes a multiple of 3 microsecond ticks. For a PD2-MF-xx-333/16H board, this becomes 12 microseconds.

For example, assume that  $N$  scans of the channel vector compose a frame of data. Assume also that you have a PD2-MF-xx-333/16H board with a **Channel vector** value of [1 2 3 4] and a **Mux settling time vector (slow bit)** value of [0 1 0 1]. For the PD2-MF-xx-333/16H board, two of the of the acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Block samptime** value of .001 second, if  $N$  is greater than 38, the data will not be acceptable. If you set all slow bits to 0,  $N$  can have a value of up to 83. The software detects the limit at  $N = 83$ , but is not aware of the extra board dependent delay.

---

## Range

From the list, select one of the following voltage ranges. The value you select applies to all channels.

- +-10 Volts
- +-5 Volts

# UEI PDXI-MFS 16-Bit Series Frame Analog Input

---

- 0-10 Volts
- 0-5 Volts

## Frame size

Enter the number of samples per channel to return as a frame of data. Due to constraints in the hardware architecture, the total number of samples in a frame of data cannot exceed 1024. The relationship between the number of samples, frame size, and number of channels is

$$\text{total number of samples} = \text{FrameSize} \times \text{nChannels}$$

For example, if you specify two channels, the frame size must be less than or equal to 512.

## Output format

From the list, select either Frame or Vector.

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

## Scan clock source

Select Internal or External to identify the clock source for the frame scans.

## Scan time

Enter the time (in seconds) between scans as the interval.

## Frame time

Enter the interval (in seconds) during which the board will interrupt. The driver block evenly spaces the scans in the frame across the **Frame time** interval. If the **Frame time** interval is too short, you will receive a buffer overrun error at run time. For optimal results, experimentally increase the **Frame time**



# UEI PDXI-MFS 16-Bit Series Frame Analog Input

---

value and reiterate the process. Always rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

$$\textit{Frametime} = \textit{Framesize} \times \textit{Scantime}$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because xPC Target uses this value to set hardware parameters on the board, you cannot change the **Frame time** value at the MATLAB prompt.

---

## **Block is in an ISR**

Select this check box if the block is in an interrupt service routine (ISR). Selecting this check box forces the block sample time to -1, which enables the block to work in the ISR. If this check box is not selected, the block sample time is equal to the value of **Frame time**.

## **Slave board**

If you have multiple boards configured in a master/slave combination, select this box for the slave boards only. Leave the box unchecked for the master board. (Note that a single board configuration is considered a master board.)

# UEI PDXI-MFS 16-Bit Series Frame Analog Input

---

---

**Note** In the model, set the priority of slave blocks higher than master blocks. Right-click the slave or master block and select **Block Properties**. In the dialog, enter a value in the **Priority** property. For example, if you have multiple slave blocks, you can enter values of 2 for the slave blocks and 3 for the master block, where a lower number indicates a higher priority.

---

## Acquisition frequency

Enter the rate at which the high speed conversion clock runs. This is the clock used to acquire each scan in a frame. For example, with a PD2-MF-xx-333/16H board, the maximum clock is 333 kHz. At that rate, the second channel in a scan will be acquired 3 microseconds after the first one in every scan of the frame.

Specify -1 to select the maximum rate available for the specified board.

## DMA burst size

From the list, select either 32, 16, or 8. The DMA engine transfers data in packets of the burst size or less. Between each burst, a new bus arbitration cycle must complete. The best performance is achieved with higher **DMA burst size** values, with 32 being highest. However, in some architectures and with some PCI bus adaptors, a **DMA burst size** of 32 might cause problems. For example, it might lock up the target machine when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the target application. If the DMA works at 8, select 16 to increase performance and try again.

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

# UEI PDXI-MFS 16-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 16-Bit Series Analog Output (D/A)

---

**Purpose** PDXI-MFS 16-Bit Series Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the output channels between 1 and 16. For example, to use the first and second analog output (D/A) channels, enter

[1 2]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for PD2-AO and PDXI-AO series boards is 8, 16, or 32 depending on the specific board type. For all other boards series the maximum channel number is 2.

**Reset vector**  
The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

**Initial value vector**  
The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output

# UEI PDXI-MFS 16-Bit Series Analog Output (D/A)

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 16-Bit Series Digital Input

---

**Purpose** PDXI-MFS 16-Bit Series Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

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

**PCI Slot (-1:autosearch)**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

# UEI PDXI-MFS 16-Bit Series Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-MFS 16-Bit Series Digital Output

---

**Purpose** PDXI-MFS 16-Bit Series Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Channel vector**  
Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

**Reset vector**  
The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

**Initial value vector**  
The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output



# UEI PDXI-MFS 16-Bit Series Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-AO Series

---

## Board Series

UEI PD2-AO Series

## General Board Series Description

The PD2-AO series boards have 8, 16, 32, or 96 analog output (D/A) channels (16-bit) with a maximum sample rate of 100 kHz per channel. They also have 8 digital input and 8 digital output lines.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-AO Analog Output (D/A)
- UEI PD2-AO Digital Input
- UEI PD2-AO Digital Output

## Board Characteristics

Board type	PD2-AO-8/16 PD2-AO-16/16 PD2-AO-32/16 PD2-AO-96/16
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PD2-AO Analog Output (D/A)

**Purpose** PD2-AO Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels between 1 and 8, 1 and 16, 1 and 32, or 1 and 96. For example, to use the first and second analog output (D/A) channels, enter

```
[1 2]
```

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for PD2-AO series boards is 8, 16, 32, or 96 depending on the specific board type.

Note, the UEI PD2-AO 96 channel board only works if you specify a contiguous group of channels. These channels also must be listed in order. If you enter a non-contiguous channel list in the **Channel** vector parameter, no output will be seen from the board.

As a workaround, use a contiguous specification like:

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

to see the output on all five channels. You can attach a ground or other appropriate signal to the channels that you do not want

# UEI PD2-AO Analog Output (D/A)

---

to use. For example, if you only want to use channels 1, 3, and 5, attach a ground to 2 and 4.

## **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

## **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** PD2-AO Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

## UEI PD2-AO Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** PD2-AO Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[ 1 3 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PD2-AO Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



## Board Series

UEI PD2-DIO-64

## General Board Series Description

The PD2-DIO-64 series PCI boards have 64 digital I/O channels in 16-bit ports, three counters/timers, two enhanced synchronous serial interface (ESSI) ports, and four high-speed (100 ns) IRQ lines.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-DIO-64 Digital Input
- UEI PD2-DIO-64 Digital Output

xPC Target only supports the digital I/O channels for this board series. It does not support the additional functionality.

## Board Characteristics

Board type	PD2-DIO-64 PD2-DIO-64ST PD2-DIO-64CT PD2-DIO-64TS
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PD2-DIO-64 Digital Input

---

**Purpose** PD2-DIO-64 Digital Input block

**Library** xPC Target Library for UEI

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

**Note** Note that each block of 16 ports must be input or output.

**Block Parameters** **Board type**  
Select the specific board type from the list provided.

**Port (first channel of port)**  
From the list, select the first channel of the port. Each port is 16 bits. This value plus the **Channel vector** value appear on the terminal block.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

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

**PCI Slot (-1:autosearch)**  
If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-DIO-64 Digital Output

---

**Purpose** PD2-DIO-64 Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Port (first channel of port)**

From the list, select the first channel of the port. Each port is 16 bits. This value and the **Channel vector** value appear on the terminal block.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

### **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. This parameter contains values of 0 or 1. It controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

## **Initial value vector**

The initial value vector contains the initial voltage values for the digital output channels. 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. This parameter contains the initial values (0 or 1) for the output channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-DIO-128

---

## Board Series

UEI PD2-DIO-128

## General Board Series

The PD2-DIO-128 series PCI boards have 128 digital I/O channels in 16-bit ports, three counters/timers, two enhanced synchronous serial interface (ESSI) ports, and four high-speed (100 ns) IRQ lines.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-DIO-128 Digital Input
- UEI PD2-DIO-128 Digital Output

xPC Target only supports the digital I/O channels for this board series. It does not support the additional functionality.

## Board Characteristics

Board type	PD2-DIO-128 PD2-DIO-128ST
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

**Purpose** PD2-DIO-128 Digital Input block

**Library** xPC Target Library for UEI

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

**Note** Note that each block of 16 ports must be input or output.

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Port (first channel of port)**

From the list, select the first channel of the port. Each port is 16 bits. This value plus the **Channel vector** value appear on the terminal block.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[ 1 3 ]

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

- 1

# UEI PD2-DIO-128 Digital Input

---

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# UEI PD2-DIO-128 Digital Output

**Purpose** PD2-DIO-128 Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Port (first channel of port)**

From the list, select the first channel of the port. Each port is 16 bits. This value and the **Channel vector** value appear on the terminal block.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

### **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. This parameter contains values of 0 or 1. It controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

# UEI PD2-DIO-128 Digital Output

---

## **Initial value vector**

The initial value vector contains the initial voltage values for the digital output channels. 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. This parameter contains the initial values (0 or 1) for the output channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board Series

UEI PD2-DIO-128i

## General Board Series

The PD2-DIO-128i series PCI boards have 64 optoisolated digital input and 64 optoisolated digital output ports in 16-bit ports.

xPC Target supports this series of boards with these driver blocks:

- UEI PD2-DIO-128i Digital Input
- UEI PD2-DIO-128i Digital Output

## Board Characteristics

Board type	PD2-DIO-128i
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PD2-DIO-128i Digital Input

---

**Purpose** PD2-DIO-128i Digital Input block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Port (first channel of port)**

From the list, select the first channel of the port. Each port is 16 bits. This value plus the **Channel vector** value appear on the terminal block.

The input and output ports on the PD2-DIO-128i are completely separate. There is no conflict if you use the same port numbers as input and output.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PD2-DIO-128i Digital Output

---

**Purpose** PD2-DIO-128i Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Port (first channel of port)**

From the list, select the first channel of the port. Each port is 16 bits. This value and the **Channel vector** value appear on the terminal block.

The input and output ports on the PD2-DIO-128i are completely separate. There is no conflict if you use the same port numbers as input and output.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

### **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. This parameter contains values of 0 or 1. It

controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector contains the initial voltage values for the digital output channels. 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. This parameter contains the initial values (0 or 1) for the output channels. The channels are set to these 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDL-DIO-64

---

## Board Series

UEI PDL-DIO-64

## General Board Series Description

The PDL-DIO-64 series PCI lab boards have 64 digital I/O channels in 16-bit ports, three counters/timers, two enhanced synchronous serial interface (ESSI) ports, and four high-speed (100 ns) IRQ lines.

xPC Target supports this series of boards with these driver blocks:

- UEI PDL-DIO-64 Digital Input
- UEI PDL-DIO-64 Digital Output

xPC Target only supports the digital I/O channels for this board series. It does not support the additional functionality.

## Board Characteristics

Board type	PDL-DIO-64 PDL-DIO-64ST PDL-DIO-64CT PDL-DIO-64TS
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes



**Purpose** PDL-DIO-64 Digital Input block

**Library** xPC Target Library for UEI

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

**Note** Note that each block of 16 ports must be input or output.

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Port (first channel of port)**

From the list, select the first channel of the port. Each port is 16 bits. This value plus the **Channel vector** value appear on the terminal block.

### **Channel vector**

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[ 1 3 ]

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# UEI PDL-DIO-64 Digital Input

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** PDL-DIO-64 Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## Block Parameters

### Board type

Select the specific board type from the list provided.

### Port (first channel of port)

From the list, select the first channel of the port. Each port is 16 bits. This value and the **Channel vector** value appear on the terminal block.

### Channel vector

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

```
[1 3]
```

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

### 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. This parameter contains values of 0 or 1. It controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

# UEI PDL-DIO-64 Digital Output

---

## **Initial value vector**

The initial value vector contains the initial voltage values for the digital output channels. 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. This parameter contains the initial values (0 or 1) for the output channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

## Board Series

UEI PDXI-DIO-64

## General Board Series Description

The PDXI-DIO-64 series PXI boards have 64 digital I/O channels in 16-bit ports, three counters/timers, two enhanced synchronous serial interface (ESSI) ports, and four high-speed (100 ns) IRQ lines.

xPC Target supports this series of boards with these driver blocks:

- UEI PDXI-DIO-64 Digital Input
- UEI PDXI-DIO-64 Digital Output

xPC Target only supports the digital I/O channels for this board series. It does not support the additional functionality.

## Board Characteristics

Board type	PDXI-DIO-64 PDXI-DIO-64ST PDXI-DIO-64CT PDXI-DIO-64TS
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PDXI-DIO-64 Digital Input

---

**Purpose** PDXI-DIO-64 Digital Input block

**Library** xPC Target Library for UEI

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

**Note** Note that each block of 16 ports must be input or output.

**Block Parameters** **Board type**  
Select the specific board type from the list provided.

**Port (first channel of port)**  
From the list, select the first channel of the port. Each port is 16 bits. This value plus the **Channel vector** value appear on the terminal block.

**Channel vector**  
Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

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

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-DIO-64 Digital Output

---

**Purpose** PDXI-DIO-64 Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

**Board type**  
Select the specific board type from the list provided.

**Port (first channel of port)**  
From the list, select the first channel of the port. Each port is 16 bits. This value and the **Channel vector** value appear on the terminal block.

**Channel vector**  
Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 0, which is the same as the hardware label.

**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. This parameter contains values of 0 or 1. It controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.



## **Initial value vector**

The initial value vector contains the initial voltage values for the digital output channels. 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. This parameter contains the initial values (0 or 1) for the output channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# UEI PDXI-AO Series

---

## Board Series

UEI PDXI-AO Series

## General Board Series Description

The PDXI-AO series boards have 8, 16, or 32 analog output (D/A) channels (16 bit) with a maximum sampling rate of 100 kHz per channel. They also have 8 digital input lines, and 8 digital output lines.

xPC Target supports this series of boards with these driver blocks:

- UEI PDXI-AO Analog Output (D/A)
- UEI PDXI-AO Digital Input
- UEI PDXI-AO Digital Output

## Board Characteristics

Board type	PDXI-AO-8/16 PDXI-AO-16/16 PDXI-AO-32/16 PDXI-AO-96/16
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# UEI PDXI-AO Analog Output (D/A)

**Purpose** PDXI-AO Analog Output block

**Library** xPC Target Library for UEI

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

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels between 1 and 16. For example, to use the first and second analog output (D/A) channels, enter

[ 1 2 ]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number for PD2-AO and PDXI-AO series boards is 8, 16, 32, or 96 depending on the specific board type. For all other boards series the maximum channel number is 2.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector and should contain the initial voltage values for the output

## UEI PDXI-AO Analog Output (D/A)

---

channels. The channels are set to these 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.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** PDXI-AO Digital Input block

**Library** xPC Target Library for UEI

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

## Block Parameters

### Board type

Select the specific board type from the list provided.

### Channel vector

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all boards except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### Sample time

Enter the base sample time or a multiple of the base sample time.

### PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with

## UEI PDXI-AO Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** PDXI-AO Digital Output block

**Library** xPC Target Library for UEI

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low > 0.5 = TTL high

## **Block Parameters**

### **Board type**

Select the specific board type from the list provided.

### **Channel vector**

Enter a vector of numbers to specify the output channels. For example, to use the first and third digital output channels, enter

[1 3]

The channel numbers can occur in any order. Number the channels beginning with 1 even if the board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 for all board types except for the PD2-AO and PDXI-AO series. For those boards, the maximum channel number is 8.

### **Reset vector**

The reset vector must be the same length as the channel vector. It contains values of 0 or 1. This parameter controls the behavior at model termination. A value of 1 causes the corresponding channel to be reset to the value specified in the initial value vector. A value of 0 causes the channel to remain at the last value attained while the model was running.

### **Initial value vector**

The initial value vector must be the same length as the channel vector. It contains the initial values (0 or 1) for the output

# UEI PDXI-AO Digital Output

---

channels. The channels are set to these 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.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```



# xPC Target Support for Vector CANape

---

xPC Target interfaces the target PC to the Vector CAN Application Environment (CANape) (<http://www.vector-cantech.com>) using the Universal Calibration Protocol (XCP). This chapter includes the following sections:

Vector CANape (p. 36-2)

Third-party monitoring system that you can use to work with xPC Target data.

Configuring xPC Target and Vector CANape (p. 36-5)

How to set up the xPC Target system to work with Vector CANape.

## Vector CANape

You can use a target PC as an electronic control unit (ECU) for a Vector CANape system. Using a target PC in this way, a Vector CANape system can read signals and parameters from a target application running on the target PC. To support this feature, xPC Target provides an XCP protocol component. The xPC Target XCP component translates XCP commands to standard xPC Target equivalents.

---

**Note** This chapter describes how to configure xPC Target and Vector CANape to work together. It also assumes that you are familiar with the Vector CANape product family. See <http://www.vector-cantech.com> for further information about the Vector CANape products.

---

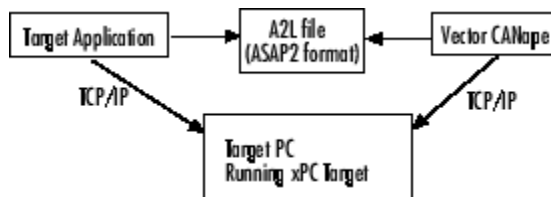
xPC Target works with Vector CANape, Version 5.00.20 and 5.00.30. To enable a target PC to work with Vector CANape, you need to

- Configure Vector CANape to use the xPC Target XCP communication layer.
- Enable xPC Target to generate a target application that can provide data compliant with Vector CANape.
- Provide a standard RS-232 or TCP/IP physical layer between the host PC and target PC.

To support the XCP communication layer, xPC Target provides

- An xPC Target XCP conversion library (`xcpforxpc.dll`) to convert the XCP commands in the user-defined XCP communication layer to the corresponding xPC Target commands.
- A generator that produces A2L files that Vector CANape can load into the Vector CANape database. The generated file contains signal and parameter access information for the target application.

The following illustrates how xPC Target and Vector CANape interact.



## Notes on xPC Target and Vector CANape

xPC Target does not support the following for Vector CANape:

- Vector CANape data acquisition commands
- Vector CANape calibration commands
- Multiple simultaneous target connections — The network connection to the xPC Target is not shareable (thread-safe). You can only attach it to one program at a time, including
  - Any program that links to the C API library `xpcapi.dll`, such as Vector CANape and any xPC Target demo program
  - The MATLAB command-line interface to xPC Target. You can use MATLAB/Simulink while Vector CANape is attached to xPC Target, but you must unload (delete) any xPC Target objects (such as those created with the MATLAB command `xpc`, or implicitly created during code generation).

To unload any xPC Target objects, type `clear all` in the MATLAB Command Window. Alternatively, to unload xPC Target objects, but keep your workspace variables, type `clear mex` in the MATLAB Command Window.

- Vector CANape signal value selection — You might get an error concerning datatypes. This error message indicates that you need to reapply the value 255 to the parameters **MAX\_CTO** and **MAX.DTO**. See “Creating a New Vector CANape Project to Associate with a Particular Target Application” on page 36-7.

### xPC Target Parameters and the A2L File

During the model build, xPC Target generates an A2L file of signal and parameter information for the Vector CANape database file. A small number

of parameter values are common to all target PCs. Other parameter values are for application-wide settings and top-level status information. xPC Target automatically inserts entries for these parameters into the generated A2L file:

- **xAveTET** (Measurement) — Average task execution time (in seconds).
- **xExecutionTime** (Measurement) — Execution time for the target application.
- **xIsOverloaded** (Measurement) — If the target PC experiences an execution overload condition, this value is set to 0. During normal operation, this value is 1.
- **xSampleTime** (Characteristic) — Base sample time (in seconds) of the target application. This can only be modified when the target PC is stopped.
- **xStartStop** (Characteristic) — If the target PC is running, this value is set to 1. If the target PC is halted, this value is set to 0.
- **xStopTime** (Characteristic) — Stop time (in seconds) of the target application. This can only be modified when the target PC is stopped.

## Configuring xPC Target and Vector CANape

To set up xPC Target and Vector CANape, see the following sections:

- “Getting Started” on page 36-5 — Set up your model to work with Vector CANape.
- “Creating a New Vector CANape Project to Associate with a Particular Target Application” on page 36-7 — Create a new Vector CANape and associate that project with a particular target application
- “Associating an Existing Vector CANape Project with a Particular Target PC” on page 36-9 — Associate an existing Vector CANape project with one particular target PC
- “Providing A2L Files for the Vector CANape Database” on page 36-10 — Provide target application A2L files for a CANape database

### Getting Started

Set up your model to work with Vector CANape. This procedure uses the `xpcosc` model. This procedure assumes that you have already configured your model to generate xPC Target code. If you have not done so, see the Getting Started with xPC Target documentation.

- 1 In the MATLAB window, type

```
xpcosc
```

- 2 In the model Simulink window, and from the **Simulation** menu, click **Configuration Parameters**.

The Configuration Parameters dialog box is displayed for the model.

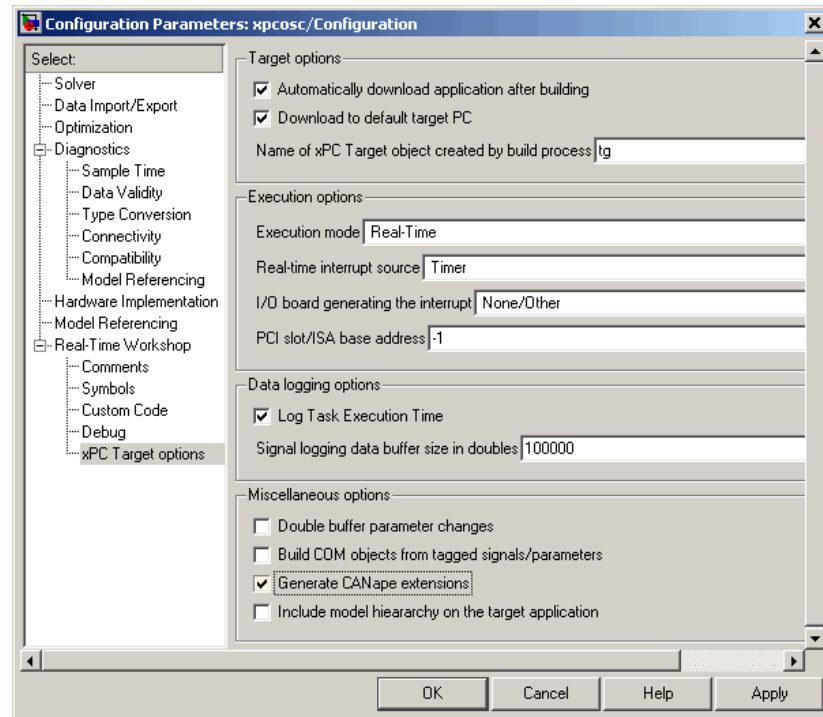
- 3 In the left pane, click the **xPC Target options** node.

The associated pane is displayed.

- 4 In the **Miscellaneous options** area, select the **Generate CANape extensions** check box.

This enables target applications to generate data, such as that for A2L, for Vector CANape.

The **Real-Time Workshop** pane should now look like the figure shown.



## 5 Build the model.

xPC Target builds the target application, including an A2L data file for the target application.

Next, copy the XCP library files (two) into the CANape binary directory. These files reside in the xPC Target directory area under the MATLAB install location. Follow these instructions:

- 1 Relative to the MATLAB root directory that you found with *matlabroot*, change directory as follows

```
cd matlabroot\toolbox\rtw\targets\xpc\api
```

- 2 Copy `xpcapi.dll` and `xcpforxpc.dll` to the binary directory `Exec` under your Vector CANape install directory. For example if your Vector CANape is installed in `C:\Applications\CANape`, then use the following command line copy commands:

```
copy xpcapi.dll C:\Applications\CANape\Exec
copy xcpforxpc.dll C:\Applications\CANape\Exec
```

## Creating a New Vector CANape Project to Associate with a Particular Target Application

This procedure describes how to create a new Vector CANape project that can communicate with an xPC Target application. If your xPC Target application is loaded and running during this process, you can test and confirm the communications configuration in step 18.

- 1 In a DOS window, create a new directory to hold your project. This can be the same directory as your xPC Target model files. For example, type

```
mkdir C:\MyProject
```

- 2 Start the Vector CANape **Create a new Project** wizard. The Windows **Start** menu contains this as a selection in the Vector CANape section.

- 3 After you create the new project, start it.

You can start the new project directly from the last screen of the create a new project wizard. After the preliminary warning, the CANape window is displayed.

- 4 In the CANape window, click **Tools > Device configuration**

The device configuration window is displayed.

- 5 In the device configuration window, click **New**.

- 6 In **Device Name**, enter a name for the device to describe your target application. For example, type

```
xPCTarget
```

Add any appropriate comments.

**7** Click **Next**.

**8** From the driver-type menu list, select **XCP**.

**9** Click **Driver configuration**.

The XCP driver settings window is displayed.

**10** Click **Extended protocol settings**.

**11** In that screen, set

- **MAX\_CTO** and **MAX\_DTO** to 255
- **Byte order** to Intel
- **Address granularity** to BYTE

You might get an error concerning datatypes. This error message indicates that you need to reapply the value 255 to the parameters **MAX\_CTO** and **MAX\_DTO**.

**12** Click **OK**.

The XCP driver settings window is displayed.

**13** In the **Transport layer** pane, from the **Interface** menu list of the, select **USER\_DEFINED**.

**14** In the **Transport layer** pane, click **Configuration**.

**15** Browse and select the file `xcpforxpc.dll` from the `CANape\Exec` directory.

**16** Click **Open**.

**17** Click **OK**.

**18** In the XCP driver settings window, verify the connection to the target PC by clicking **Test connection**. This command only succeeds if the target PC is running and connected properly. Be sure that no other host is connected to the target PC.

**19** Click **OK** and answer Yes if you are asked to update your database.



- 20** You have completed the configuration of Vector CANape for xPC Target. Continue with the rest of the Vector CANape wizard dialog as usual.

Next, provide your target application A2L file to Vector CANape. See “Providing A2L Files for the Vector CANape Database” on page 36-10. If you want to associate your Vector CANape project with a particular target PC, see “Associating an Existing Vector CANape Project with a Particular Target PC” on page 36-9.

This CANape project should now be ready to monitor and control your xPC Target. The CANape database should be populated with a comprehensive list of xPC Target application signals and parameters that are available. During target revisions, it may be necessary to manually reload the A2L that is generated by the xPC Target build process. This can be done from the CANape Database editor.

## **Associating an Existing Vector CANape Project with a Particular Target PC**

You might want to force a particular Vector CANape project to use a specific set of communications properties if either of the following situations applies:

- You are working with multiple target PCs from the same host computer.
- You are using a host computer that does not have an installed copy of MATLAB and xPC Target.

This topic assumes that

- You have set up and built your model to generate data for Vector CANape. If you have not yet done so, see “Getting Started” on page 36-5.
- You have an existing Vector CANape project directory

When Vector CANape connects to a target PC, it reads the communications parameters from the last MATLAB session that configured an xPC Target. The XCP layer searches for the xPC Target configuration file (`xpcenv.mat`, a standard MATLAB MAT file). Vector CANape automatically connects to that same target PC. (If you do not have an `xpcenv.mat` file, type the command `updatexpcenv(1)`, including the argument `1`, at the MATLAB Command Prompt to create one.)

---

**Note** Clear any MATLAB connections before opening CANape, as described in “Notes on xPC Target and Vector CANape” on page 36-3.

---

**1** In the MATLAB Command Window, type

```
xpcexplr
```

**2** Configure the communications protocols as desired for the target PC.

**3** If the target PC is connected to this computer, test the connection. In the MATLAB Command Window, type `xpctargetping` and/or `xpc`.

**4** If you are satisfied with these settings, copy the xPC Target configuration file to the desired Vector CANape project directory. For example:

```
copy C:\WINNT\Profiles\smith\Application  
Data\MathWorks\MATLAB\R14\xPCTargetPrefs C:\MyProject
```

To retrieve the actual path of `xpcenv.mat`, you can use the following MATLAB command:

```
[prefdir '\xPCTargetPrefs']
```

The Vector CANape project now has a local copy of `xpcenv.mat`. It will always connect to the same target PC, regardless of the state of MATLAB. Next, you must provide your target application A2L file to Vector CANape. See “Providing A2L Files for the Vector CANape Database” on page 36-10.

## **Providing A2L Files for the Vector CANape Database**

This topic assumes that

- You have set up and built your model to generate data for Vector CANape. If you have not yet done so, see “Getting Started” on page 36-5.
- You have created a Vector CANape project directory and know the name of that project directory.

To enable Vector CANape to load the A2L file for the model `xpcosc`:

- 1** In a DOS shell, change directory to the one that contains the A2L file from the previous procedure. For example:

```
cd D:\work\xpc
```

- 2** Look for and copy the A2L file to your Vector CANape project directory. For example:

```
copy xpcosc.a2l C:\MyProject
```

Vector CANape automatically loads the target application A2L file when it connects to the target PC.



# Versallogic

---

This chapter describes the Versallogic I/O board supported by xPC Target (<http://www.versallogic.com>).

Versallogic VSBC-6

A single-board computer with 8 signal ended analog input (A/D) channels, 16 digital I/O lines, and a watchdog timer.

## **Boards and Blocks**

**Board** Versallogic VSBC-6

**General Description** The VSBC-6 is a single board computer with 8 signal ended analog input (A/D) channels, 16 digital I/O lines, and a watchdog timer.

xPC Target supports this board with these driver blocks:

- Versallogic VSBC-6 Analog Input (A/D)
- Versallogic VSBC-6 Digital Input
- Versallogic VSBC-6 Digital Output
- Versallogic VSBC-6 Watch Dog

<b>Board Characteristics</b>	Board type	VSBC-6
	Manufacturer	Versallogic
	Bus type	N/A
	Access method	N/A
	Multiple block instance support	Yes
	Multiple board support	No

# Versallogic VSBC-6 Analog Input (A/D)

---

**Purpose** VSBC-6 Analog Input block

**Library** xPC Target Library for Versallogic

**Block Parameters**

**Channel vector**

Enter numbers between 1 and 8. This driver allows you to enter channel numbers in any order.

For example, to use the first, second, and fifth channels, enter

[1,2,5]

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

**Range vector**

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This driver allows each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 to +10 volts and the second and fifth channels are 0 to +5 volts, enter

[-10,5,5]

**Sample time**

Model base sample time or a multiple of the base sample time.



# Versallogic VSBC-6 Digital Input

**Purpose** VSBC-6 Digital Input block

**Library** xPC Target Library for Versallogic

## Scaling Input to Output

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

## Block Parameters

### Channel vector

Enter a numbers between 1 and 16 to select the number of digital input lines used. This driver allows the selection of individual digital input lines in any order.

For example, to use the first, second and fifth digital input lines, enter

[ 1,2,5]

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

### Sample time

Enter the base sample time or a multiple of the base sample time.

# Versallogic VSBC-6 Digital Output

---

**Purpose** VSBC-6 Digital Output block

**Library** xPC Target Library for Versallogic

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ± 0.5 = TTL high

## **Block Parameters**

### **Channel vector**

Enter numbers between 1 and 16 to select the number of digital output lines used. This driver allows the selection of individual digital output lines in any order.

For example, to use the first, second and fifth digital output lines, enter

[1,2,5]

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

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

**Purpose** VSBC-6 Watch Dog block

**Library** xPC Target Library for Versallogic

**Block Parameters**

**Show enable port**

Select this check box to show a digital input port on the driver block. This port indicates that the watchdog timer is enabled. The watchdog timer starts in a disabled state. If you want to use the VSBC-6 board to reboot the system when a programmable timeout occurs, select this check box.

**Show reset port**

Select this check box to show a digital input port on the driver block. A signal connected to this port resets the watchdog whenever its value exceeds 1.

**Sample time**

Enter the base sample time or a multiple of the base sample time.



# VMIC

---

This chapter describes the VMIC reflective shared memory board supported by xPC Target (<http://www.vmic.com>).

Before You Start (p. 38-2)

Guidelines for setting up structures for shared memory.

VMIC Shared Memory Structure Reference (p. 38-6)

Shared memory reference for structures.

Boards and Blocks — Alphabetical List (p. 38-13)

Description of block parameters for VMIC driver blocks.

## Before You Start

xPC Target uses a model for reflective memory (also known as shared memory) that includes Simulink blocks, for the shared memory drivers, and MATLAB structures for defining shared memory and node initialization partitions. The topics in this section are

- “Create Shared Memory Partitions” on page 38-2 — Before you begin to use VMIC shared memory blocks, define a partition structure that defines how you want to allocate shared memory.
- “Initialize Shared Memory Nodes” on page 38-4 — Before you begin to use VMIC shared memory blocks, define a node initialization structure that defines how shared memory is allocated (partitioned) and how the board is configured.

### Create Shared Memory Partitions

To use the xPC Target VMICPCI-5565 shared memory blocks to read, write, pack, or unpack data, you must define a partition structure. VMIC shared memory drivers use MATLAB structures to define shared memory partitions. A partition structure describes how you want to allocate (or partition) the shared memory. xPC Target allocates shared memory with bundles of data that are packed into memory partitions. The following VMIC blocks use shared memory partition structures:

- VMIC VMICPCI-5565 read and VMIC VMICPCI-5565 write
- Shared Memory Pack and Shared Memory Unpack

After you define the shared memory partitions, you can add VMIC shared memory driver blocks to your Simulink model. Create a shared memory structure in one of the following ways. See “Shared Memory Partition Structure” on page 38-6 for the complete list of fields for the structure.

The following description refers to the `completepartitionstruct` command.  
Type

```
help completepartitionstruct
```

for a description of the command.

- Using the `completepartitionstruct` command at the MATLAB Command Window, create a default partition structure. For example, type

```
completepartitionstruct([], '5565')
```

```
ans =
```

```
    Address: '0x0'
      Type: 'uint32'
       Size: '1'
  Alignment: '4'
   Internal: [1x1 struct]
```

- At the MATLAB Command Window, create a user-defined partition structure. The easiest way to do this is to create an M-file, partially define a structure, load that M-file into the MATLAB workspace, and supplement the resulting structure with a call to the `completepartitionstruct` function. For example

```
Partition(1).Address='0x5000';
Partition(1).Type='int8';
Partition(1).Size='10';
Partition(2).Type='uint16';
Partition(2).Size='5';
Partition(3).Type='uint8';
Partition(3).Size='1';
Partition(3).Alignment='8';
Partition(4).Type='double';
Partition(4).Size='3';
```

This example defines a partition with four segments.

- The `Address` field is optional. Only specify this field for the first segment of a partition. The elements of a partition are defined as a continuous memory block from the first address. The following segments extrapolate their addresses from the first segment. If you have fragmented memory, use multiple partitions and VMIC read/write blocks to work with the memory.
- The `Type` and `Size` fields are required for all fields in the partition structure.

- The Alignment value is optional. It is '4' by default, which forces segments that do not have alignment specifications to start on 4 byte (32 bit) boundaries. In this partition, the third segment (Partition(3)) has an alignment of '8'.
- The base addresses of subsequent segments are fully defined by the data type, size, and alignment of the preceding segment.

You can then call the `completpartitionstruct()` command to full populate the partition structure.

## Initialize Shared Memory Nodes

To use the xPC Target VMICPCI-5565 shared memory, you must define a node initialization structure. VMIC shared memory drivers use MATLAB structures to define shared memory node initialization. A node initialization structure describes the shared memory partition (see “Create Shared Memory Partitions” on page 38-2) and the VMIC board configuration, including interrupt configurations. The following VMIC block requires shared memory node initialization structures:

- 5565 init

After you define the node initialization partition, you can add VMIC shared memory driver blocks to your Simulink model. Create a shared memory structure in one of the following ways. See “Shared Memory Node Initialization Structure” on page 38-8 for the complete list of fields for the structure.

The following description refers to the `completenodestruct` command. Type

```
help completenodestruct
```

for a description of the command.



- Using the `completenodeconstruct` command at the MATLAB Command Window, create a default node initialization structure. For example, type

```
node=completenodeconstruct([], '5565')
```

```
node =
```

```
    Interface: [1x1 struct]  
    Partitions: [1x1 struct]
```

- A user-defined node structure, create with m-code or from the MATLAB Command Window and supplement the resulting structure with a call to the `completenodeconstruct` function.

## VMIC Shared Memory Structure Reference

You do not need to use all the fields of a partition or node initialization structure. However, knowing the possible structure fields will be helpful when you are creating any of the structures.

This section includes the following topics:

- “Shared Memory Partition Structure” on page 38-6 — Description of the partition node structure fields
- “Shared Memory Node Initialization Structure” on page 38-8 — Description of the node initialization structure fields

### Shared Memory Partition Structure

A shared memory partition structure has the following fields:

```

Address: '0x0'
Type: 'uint32'
Size: '1'
Alignment: '4'
Internal: [1x1 struct]

```

where

Partition Fields	Description
<b>Address</b>	Specifies the base address (in hexadecimal) of the memory partition within the node’s shared memory space. The default value is '0x0', the first location in shared memory.  Align partition addresses on 32-bit word boundaries (for example, 0x0, 0x4, 0x8, and so forth).

<b>Partition Fields</b>	<b>Description</b>
<b>Type</b>	<p>Specifies the data type of the memory segment. Specify one of the following types:</p> <ul style="list-style-type: none"> <li>• single (IEEE Single Precision)</li> <li>• double (IEEE Double Precision)</li> <li>• uint8</li> <li>• int8</li> <li>• uint16</li> <li>• int16</li> <li>• uint32</li> <li>• int32</li> <li>• Boolean (a single byte represents a boolean value)</li> </ul> <p>The default value is 'uint32'. A minimum partition size is 32 bits.</p>
<b>Size</b>	<p>Specifies the dimension and size of the memory segment. You can enter a scalar value or a value with the [m,n] format. The default value is '1'.</p> <ul style="list-style-type: none"> <li>• scalar — Treats the Size entry as the specification of the length of a non-oriented array or vector</li> <li>• [m,n] — Treats the Size entry as an array dimension. The total number of elements in this segment is m*n.</li> </ul>
<b>Alignment</b>	<p>If another partition precedes this partition, defines the byte alignment of this segment. Specify one of the following alignment values: 1, 2, 3, 4, or 8. The default value is '4'. This value forces a double word boundary alignment for all elements.</p>
<b>Internal</b>	Reserved for internal use.

## Shared Memory Node Initialization Structure

A node initialization structure has the following fields:

```
Interface: [1x1 struct]
Partitions: [1x1 struct]
```

where

Node Structure Fields	Description
<b>Interface</b>	<p>Specifies how the board is configured. The Interface structure has the following fields, three of which are structures:</p> <ul style="list-style-type: none"> <li>• Mode — Configures board registers (see “Board Mode” on page 38-8)</li> <li>• Interrupts — Enables the board to generate PCI interrupts from network events that have been broadcast from other nodes, or error conditions (see “Board Interrupts” on page 38-10)</li> <li>• NodeID — Specifies the node ID for the board (see “Board Node ID” on page 38-12)</li> <li>• Internal — Reserved for internal use</li> </ul>
<b>Partitions</b>	Stores the shared memory segments (see “Create Shared Memory Partitions” on page 38-2)

### Board Mode

The VMIC board has a number of registers that you can set through the `Interface.Mode` field. To display the board mode fields, type

```
>> node.Interface.Mode

ans =
      StatusLEDOff: 'off'
      TransmitterDisable: 'off'
```

```

DarkOnDarkEnable: 'off'
LoopbackEnable: 'off'
LocalParityEnable: 'off'
MemoryOffset: '0'
MemorySize: '64MByte'

```

Note that mode values affect the VMIC board setting of the LSR1 (Local Control and Status Register 1) and LIER (Local Interrupt Enable Register) registers. Refer to the VMIC product documentation for further details on these two registers. To monitor the status of these modes, select the **Error Status Port** check box of the VMIC 5565 read or write block.

Of particular note are the following modes:

<b>Board Modes</b>	<b>Description</b>
<b>StatusLEDOff</b>	Turns the VMIC board status LED on and off. Setting this value to 'off' turns off the LED when the xPC Target model runs, setting this value to 'on' turns on the LED when the xPC Target model runs. When the xPC Target terminates, the LED status reverses in both cases. The default value is 'off'.
<b>MemoryOffset</b>	Applies a global offset to all network data transfers coming from the VMIC board. The following table lists offset values and the resulting offset. The default value is '0'.
<b>MemorySize</b>	Specifies the minimum memory size required. The VMIC driver checks this value against the memory size of the VMIC board. A VMIC board has a memory size of either '64MByte' or '128MByte'. If you enter a size in this field that is larger than the actual VMIC board memory size, the driver will return an error. The default value is '64MByte'.

This table lists the values for MemoryOffset:

Value	Offset Produced
'0'	0
'1'	0x4000000
'2'	0x8000000
'3'	0xC000000

### Board Interrupts

The VMIC board can generate PCI interrupts in response to network events that have been broadcast from other nodes, or error conditions. For example, you can configure two xPC Target Simulink models, one as master, and one as a slave of the broadcast node in the master xPC Target model. In such a configuration, the broadcast node interrupt triggers the model's time steps.

To display the interrupt mode fields, type

```
>> node.Interface.Interrupts
```

```
ans =
```

```

    LocalMemoryParity: 'off'
    MemoryWriteInhibited: 'off'
      LatchedSyncLoss: 'off'
        RXFifoFull: 'off'
          RXFifoAlmostFull: 'off'
            BadData: 'off'
              PendingInit: 'off'
                RoguePacket: 'off'
                  ResetNodeRequest: 'off'
                    PendingInt3: 'off'
                      PendingInt2: 'off'
                        PendingInt1: 'off'

```

Each field corresponds to a bit in the LIER register of the VMIC board. Each bit enables the specified interrupt source on the VMIC board. Refer to the VMIC product documentation for further details on this register.

To enable a node to generate a network interrupt source, add the 5565 broadcast block to a model (for example, the master model). This block issues network interrupts at the model sample rate. Correspondingly, to enable other nodes of the network (for example, the slaves) to accept broadcast interrupts from the model, configure the slave model to expect the broadcast interrupt.

The following procedure describes how to configure an entire xPC Target model to accept a broadcast interrupt from a VMIC board. See VMIC VMICPCI-5565 broadcast for a description of the **Interrupt parameter** value that the xPC Target model expects.

- 1 From the MATLAB Command Window, type

```
getxpcpci
```

This command lists board information for all installed PCI devices that xPC Target knows about.

- 2 Find the IRQ specified for the VMIC board.

This is the interrupt source number you need to specify in the **xPC target code generation options** field in step of the following procedure.

- 3 Edit your M-file and add a line like the following.

```
node.Interface.Interrupts.PendingInt1='on'
```

This line directs the model to expect an interrupt. It assumes that the value of the 5565 broadcast block **Interrupt parameter** is 1.

- 4 From the MATLAB Command Window, type the name of your Simulink model.

The Simulink model appears.

- 5 From the **Simulation** menu, click **Simulation Parameters**.

- 6 Click the **Real-Time Workshop** tab.

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

- 8 Ensure that the **Execution mode** field is set to Real-Time.

- 9 Click the **Real-Time interrupt source** list.
- 10 Select the interrupt number to which the board is set (from step ).
- 11 Click the **I/O board generating the interrupt** list and select VMIC-5565 from the list.
- 12 Click **OK**.

---

**Note** If you have a larger model, and you want to localize control of the interrupt within that model, use the IRQ Source block from the Asynchronous Event sublibrary.

---

### Board Node ID

The jumpers of the VMIC board specify the board node ID. Correspondingly, you can also configure the VMIC block with the board node ID using the `Interface.NodeID` field. Enter values according to the following:

NodeID Value	Description
'any'	Allows the VMIC driver to work with any VMIC node regardless of the VMIC board node ID jumper setting
value from '0' to '255'	Specifies the particular VMIC node that the driver must look for. If this value does not match the jumpered value on the VMIC board, the driver returns an error.

The default value of 'any' suffices in most instances. However, you might want to specify a particular NodeID value if you have multiple VMIC boards in your system and you want to identify the driver for a particular node.



## **Boards and Blocks – Alphabetical List**

VMIC VMICPCI-5565  
VMIC VMICPCI-5565 broadcast  
VMIC VMICPCI-5565 init  
VMIC VMICPCI-5565 read  
VMIC VMICPCI-5565 write

# VMIC VMICPCI-5565

---

**Board** VMIC VMICPCI-5565

**General Description** The VMICPCI-5565 is a high-speed fiber optic reflective memory. It can also generate/broadcast interrupts. xPC Target uses this board as part of the shared memory network that you can use to exchange data between computer nodes.

xPC Target supports this board with these driver blocks:

- VMIC VMICPCI-5565 broadcast
- VMIC VMICPCI-5565 init
- VMIC VMICPCI-5565 read
- VMIC VMICPCI-5565 write

For information about the Change endianness block, see Byte Reversal/Change Endianness.

<b>Board Characteristics</b>	Board name	VMICPCI-5565
	Manufacturer	VMIC
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

**Purpose** VMIC VMICPCI-5565 broadcast block

**Library** xPC Target Library for VMIC

**Note** The 5565 broadcast block generates a network interrupt that other boards in the shared memory network can detect.

The 5565 broadcast block has two inputs:

- **En** — This input port is a Boolean input that can enable or disable the transmission of a network interrupt. For continuous interrupts, connect this input port to a Constant block with a Boolean value of true (1). If you want to enable interrupts only part of the time, you can use another kind of input, such as a pulse generator.
- **Data** — This input port allows you to transmit a data value (uint32) with the interrupt. Note that xPC Target does not support receiving this data value.

## Block Parameters

### Target Node

Enter the node ID of the node to which to broadcast the interrupt. Enter -1 to broadcast the interrupt to all the nodes in the shared memory network. Note that you cannot broadcast the interrupt to any other subset of nodes.

### Interrupt Number

Enter the VMIC shared IRQ number. This value is the special interrupt number that VMIC boards send between each other (see the LIER register bit descriptions in the VMIC product documentation). This value must match the value of X in a line like the following, which you specify for the receiving end of the shared memory network (see “Board Interrupts” on page 38-10).

```
node.Interface.Interrupts.PendingIntX
```

### Sampletime

Enter the base sample time or a multiple of the base sample time.

# VMIC VMICPCI-5565 broadcast

---

## PCI Slot (-1:autosearch)

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** VMIC VMICPCI-5565 init block

**Library** xPC Target Library for VMIC

**Note** Before you begin to configure these block parameters, be sure that you have a predefined node initialization structure. See “Initialize Shared Memory Nodes” on page 38-4.

Each model that uses shared memory must have one 5565 init block for every VMIC board in the system.

**Block Parameters**

**Node struct**

Enter the name of the predefined node initialization structure.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# VMIC VMICPCI-5565 read

---

**Purpose** VMIC VMICPCI-5565 read block

**Library** xPC Target Library for VMIC

**Note** Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. The 5565 read block requires this structure to specify how Simulink signal values are mapped in the shared memory. It also uses this structure to determine the total size and address of the shared memory. See “Create Shared Memory Partitions” on page 38-2.

## **Block Parameters**

### **Partition struct**

Enter the name of the predefined shared memory partition structure. The block uses this structure to specify how Simulink signal values map into shared memory.

### **Sampletime**

Enter the base sample time or a multiple of the base sample time.

### **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

### **Error Status Port**

Select this check box to monitor the status of the VMIC board LISR register modes.

**Purpose** VMIC VMICPCI-5565 write block

**Library** xPC Target Library for VMIC

**Note** Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. The 5565 write block requires this structure to specify how Simulink signal values are mapped in the shared memory. It also uses this structure to determine the total size and address of the shared memory. See “Create Shared Memory Partitions” on page 38-2.

**Block Parameters**

**Partition struct**

Enter the name of the predefined shared memory partition structure. The block uses this structure to specify how Simulink signal values map into shared memory.

**Sampletime**

Enter the base sample time or a multiple of the base sample time.

**PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Error Status Port**

Select this check box to monitor the status of the VMIC board LISR register modes.





# Miscellaneous Blocks

---

This chapter describes miscellaneous blocks supported by xPC Target:

Asynchronous Event Support  
(p. 39-2)

xPC Target includes support for asynchronous events. These events are triggered by a hardware interrupt asynchronously to normal execution.

Blocks — Alphabetical List (p. 39-6)

Description of block parameters for miscellaneous driver blocks.

## Asynchronous Event Support

xPC Target includes support for asynchronous events. These events are triggered by a hardware interrupt asynchronously to normal execution. Some I/O boards raise interrupts that the CPU can use to interrupt the normal execution of code and jump to another sections of code called an Interrupt Service routine (ISR).

This section includes the following topics:

- “Adding an Asynchronous Event” on page 39-2 — Add an Async IRQ Source block to your Simulink model
- Async IRQ Source — Reference for block parameters
- Async Rate Transition — Reference for block parameters
- Async Buffer Write and Read — Reference for block parameters
- “Asynchronous Interrupt Examples” on page 39-5 — Examples for data transfer with a Transition block, buffered data transfer with Read/Write blocks, and interrupt CAN communication with PCI and PC/104 boards

### Adding an Asynchronous Event

When developing a model in Simulink that runs in the xPC Target environment, an Interrupt Server Routine (ISR) is modeled by using a Function-Call Subsystem. Additionally, you need to add an IRQ Source block connected to the Function-Call Subsystem block. This subsystem is then executed when an interrupt occurs and the CPU is ready to accept it.

After you install an I/O board with interrupt support into your target PC, you can add xPC Target asynchronous blocks to your Simulink model.

- 1 In the MATLAB Command Window, type

```
xpcLib
```

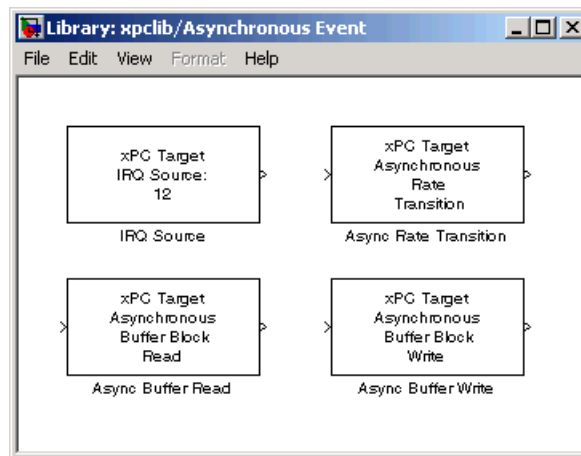
The xPC Target Library opens.

- 2 Double-click the Asynchronous Event group block.

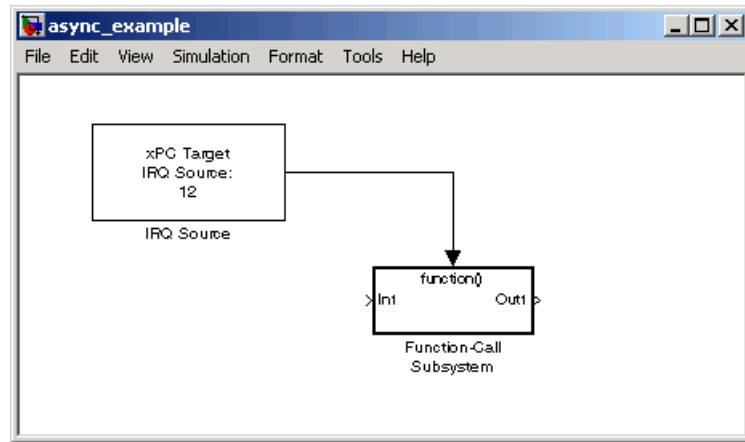


Asynchronous  
Event

The Library: xpclib/Asynchronous Event window opens.



- 3** Drag-and-drop the xPC Target IRQ block in your Simulink model and connect the output to this block to the input of a Function-Call Subsystem. For more information on Function-Call subsystems, see the Simulink and Real-Time Workshop documentation.



In the setup shown above, the CPU executes the contents of the Function Call-Subsystem whenever IRQ 5 occurs.

**4** Double-click the IRQ Source block.

The Block Parameters: IRQ Source dialog box opens.

**5** From the IRQ line number list, choose **1, 2, . . . , 15**. To determine the available IRQ line numbers on the target PC, use the function `getxpcpci`.

**6** Select or deselect the **Allow preemption of function call subsystem** check box.

**7** From the I/O board generating the interrupt list, select an interrupt board.

**8** In the PCI slot box, enter the PCI slot number or enter -1 to let xPC Target determine the number.

**9** Click **OK**.

For more information about the IRQ Source block, see Async IRQ Source.

If you need to transfer data from your ISR, add an Async Transition Block or Async Read/Write blocks to your Simulink model. See Async Rate Transition, Async Buffer Write and Read, and “Asynchronous Interrupt Examples” on page 39-5.

If you are using a CAN field bus with interrupts, see “Asynchronous Interrupt Examples” on page 39-5.

## Asynchronous Interrupt Examples

xPC Target provides several example models. If you installed MATLAB in the default location, these models are located in

```
C:\MATLAB\toolbox\rtw\targets\xpc\xpcdemos
```

To access any of these models, in the MATLAB Command Window, type the name of the model. Each model contain annotations documenting its purpose, and should serve as an example of how to use these blocks.

- `xpcasynbuffer` — Model using an external TTL signal to trigger and interrupt on the PCI-CTR05 board. Data exchange between an asynchronous task and a monotonic task using **Async Buffer Read/Write** blocks.
- `xpcasynctrans` — Model using an external TTL signal to trigger and interrupt. on the PCI-CTR05 board. Data exchange between and asynchronous task and a rate monotonic task using an **Async Rate Transition** block.
- `xpccanintpc104` — Model using interrupt driven CAN I/O communication with the CAN-AC2-104 board.
- `xpccanintpci` — Model using interrupt driven CAN I/O communication with the CAN-AC2-PCI board.

## **Blocks – Alphabetical List**

# Async Buffer Write and Read

---

<b>Purpose</b>	Async Buffer Write and Read blocks
<b>Library</b>	xPC Target Library for Asynchronous Event
<b>Description</b>	<p>These blocks provide double buffering of data between the ISR and the model which executes rate-monotonically in real-time. Always use these blocks in pairs with an Async Buffer Write Block leading into an Async Buffer Read block. The Async Buffer Write Block has to be part of the ISR, and the Async Buffer Read block is outside the ISR.</p> <p>Use Async Buffer Write and Read blocks for a more secure method of transferring data. Unlike the rate transition block, the data from one buffer is not copied to the second buffer. Instead, interrupts are disabled, and pointers to the buffers are swapped. This method produces the smallest time the interrupts are disabled and protects against data corruption caused by overwriting partially copied buffers.</p>
<b>Block Parameters</b>	<p><b>Sample time</b> Enter a base sample time or a multiple of the base sample time.</p>
<b>See Also</b>	“Asynchronous Event Support” on page 39-2

# Async IRQ Source

---

<b>Purpose</b>	Async IRQ Source block
<b>Library</b>	xPC Target Library for Asynchronous Event
<b>Description</b>	The main block that notifies Simulink and xPC Target that a particular Function-Call Subsystem should be treated as an ISR is the IRQ Source block. This block is actually a virtual block and does not exist at model execution time. However, the model initialization code sets things up with the CPU to execute the ISR when the proper interrupt occurs.
<b>Block Parameters</b>	<p><b>IRQ line number</b></p> <p>Select the IRQ line number you are using for this block. This depends on the characteristics of your hardware. You may need to query the PCI bus on the target PC to find what IRQ the PCI bus assigned to your hardware. Use the function <code>getxpcpci</code>.</p> <p>Valid IRQ numbers are between 3 and 15.</p> <p><b>Allow preemption of function-call subsystem</b></p> <p>Normally, while the ISR is executing, another interrupt will not cause re-execution of the ISR (That is the ISR will not <i>preempt</i> itself). If this check box is checked, the ISR will interrupt itself.</p> <p><b>I/O board generating the interrupt</b></p> <p>For many I/O boards, you need to set up the board properly to generate the interrupt. You might also need to set up board specific features at the beginning and/or end of an ISR. Select the correct board that you intend to use from the drop-down list.</p> <p><b>PCI slot (-1: autosearch)</b></p> <p>If only one board of this type is in the target PC, enter</p> <p style="padding-left: 40px;">-1</p> <p>to automatically locate the board.</p> <p>If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of the board associated with</p>



this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**See Also**

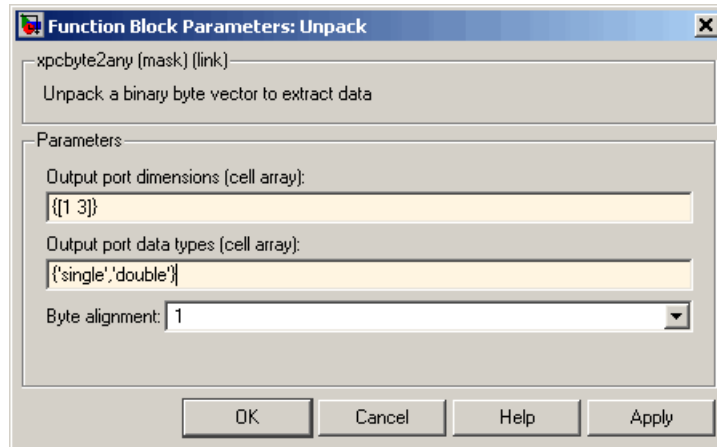
“Asynchronous Event Support” on page 39-2

# Async Rate Transition

---

<b>Purpose</b>	Async Rate Transition block
<b>Library</b>	xPC Target Library for Asynchronous Event
<b>Description</b>	<p>Use the Asynchronous Rate Transition block to double buffer data between the function call subsystem and the rest of the model which executes rate-monotonically in real-time.</p> <p>Normally, the interrupt service routine writes to the first buffer. When the next model step executes, the first buffer is copied to the second buffer and its value is used for model calculations.</p> <p>If a second interrupt occurs while the buffer is being copied, data is corrupted. This corruption happens when part of the data is copied from the first buffer, the interrupt occurs and writes over the entire first buffer, and then the transition block continues the copy operation from the first buffer that now has data from the second interrupt.</p> <p>To prevent possible data corruption, use Async Buffer Write and Read blocks. See Async Buffer Write and Read blocks.</p>
<b>Block Parameters</b>	<p><b>Sample time</b> Enter a base sample time or a multiple of the base sample time.</p>
<b>See Also</b>	“Asynchronous Event Support” on page 39-2

<b>Purpose</b>	From File
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	<p>The xPC Target From File block reads data from a file on the target PC hard disk and distributes that data in chunks every sample time. As the xPC Target kernel on the target PC reads the file data, it writes that data into a software buffer whose size is user-defined. The From File block then reads the data from this buffer in real time. One example where you might want to use the From File block is if you want to use externally-acquired data (data from a file) to drive a model.</p> <p>The From File block distributes the data as a sequence of bytes. To use these data bytes as input to a model, convert the data into one or more signals. To do so, use the UDP byte Unpack block. The UDP Unpack block outputs data in various Simulink data types. For example, assume data in your file represents a single precision scalar and a double precision vector of width 3. To convert data of this type, set up the block to output every sample time:</p> <pre>28 bytes (1 * sizeof('single') + 3 * sizeof('double'))</pre> <p>This can then be converted into the appropriate signals by the Byte Unpacking block.</p>



See the following topics:

- “File Format” on page 39-12 — Describes the target PC source file format
- “Block Parameters” on page 39-13 — Describes the block parameters for the From File block

## File Format

Before you use a target PC file as the source for the From File block, ensure that the data in the file is formatted properly. The file format is a concatenation of the different data elements for one time step, followed by the next time step, and so on.

For example, assume that the data in your file is the same as that in the preceding example. Assign a variable to each component, for example,

- a — single precision value
- b — double precision vector of 3

Assume, also, that there are N time steps worth of data. The array dimension for a and b are then

- `size(a)` — [1, N]
- `size(b)` — [3, N]

In sequence, write out the data like the following to create the file.

```
a(1, 1)    4 bytes
b(:, 1)   24 bytes
a(1, 2)    4 bytes
b(:, 2)   24 bytes
...
...
a(1, N)    4 bytes
b(:, N)   24 bytes
```

If you already have the data as MATLAB variables, use the `xpcbytes2file` function to create the file on the host PC before transferring it to the target PC. This function has the following syntax:

```
xpcbytes2file(filename, var1, ... varn)
```

where

- `filename` — Specify the name of the data file from which the From File block distributes data
- `var1, ... varn` — Specify the column of data to be output to the model.

## Block Parameters

### Filename

Enter the name of the target PC file that contains the data.

### Output port width

Enter the size, in bytes, of the data to be distributed each sample time.

## **Buffer size**

Enter the size of the software FIFO, in bytes. The xPC Target kernel fills this FIFO with the data to be input to the model. The From File block empties this FIFO as it inputs the data to the model.

This parameter should ideally be

- Much larger than **Output port width**
- At least several times the disk read size

If your model has varying Task Execution Times (because of multitasking or because of the use of conditionally executed subsystems), increasing this parameter value helps prevent the real-time application from emptying the buffer faster than the background task can compile it, causing eventual saturation.

## **Disk read size**

Enter the number of bytes to read when necessary. To understand this parameter, assume the following default values:

- **Buffer size** is 2000
- **Disk read size** is 512
- **Output port width** is 8

This means that the data buffer is of size 2000. This buffer is initially full.

Each time the block executes, eight bytes are output to the model, and the number of bytes in the buffer decreases by eight. Each time the number of free bytes in the buffer goes to 512 or higher, the xPC Target kernel attempts to read 512 bytes from the xPC Target data file to fill the buffer.

Setting this parameter to another value, for example 1024, causes the From File block to wait until 1024 bytes are free before attempting the next read.

For efficiency, set this value to a multiple of 512 (a disk sector is 512 bytes).

### **When reaching EOF**

Select the behavior of the block for when you run the application beyond when you have data in the file. Select

- **Hold last output** — Stops reading and stops the output at the last value
- **Seek to beginning** — Returns to the beginning of the file and starts reading the data (this option results in periodic data)

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

# From xPC Target

---

<b>Purpose</b>	Data from target PC
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	This block behaves like a source. Its output is usually connected to the input of a display device.

---

**Note** The use of From xPC Target blocks requires a connection between the host and target PC. If there is no connection between the host and target PC, operations such as opening a model that contains these blocks or copying these blocks within or between models, will take significantly longer than normal.

---

Some notes on the From xPC Target block behavior:

- To highlight a signal line that a From xPC Target block refers to, double-click the From xPC Target block.
- If the From xPC Target block has not yet been configured, double-clicking the From xPC Target block has no effect.
- To edit the From xPC Target block parameters, right-click the block and select **Mask Parameters**.

## Block Parameters

### **xPC Target application name**

The function `xpcsliface` automatically enters a name entry for this parameter. It is the same name as the Simulink model that xPC Target uses to build the target application.

### **Signal name (block name)**

The function `xpcsliface` automatically enters a name entry for this parameter.



### **Observer sample time**

The function `xpcsliface` automatically enters the sample time for the Simulink block with this signal. It can be equal to the model base sample time or a multiple of the base sample time.

### **Use default target PC**

Selecting this option directs the From xPC Target block to communicate to the default target PC. This assumes that you configured a default target PC through the xPC Target Explorer (see sections “Serial Communication” and “Network Communication” if you have not). By default, this check box is selected.

### **Specify target name**

If you deselect the **Use default target PC** check box, this field is displayed. Enter the name of the configured target PC.

### **See Also**

“Creating a Custom Graphical Interface” of the xPC Target User’s Guide

# I/O Port Read

---

**Purpose** I/O Port Read

**Library** xPC Target Library for Misc.

**Description** Intel 80x86 and compatible processors have a special address space reserved for I/O devices.

**Example** To access the special address space and communicate directly to a device, xPC Target provides the I/O Port Read and I/O Port Write blocks. These blocks enable the transfer of data from and to the I/O ports (see also I/O Port Write).

**1** Double click the I/O Port Read block.

The Block Parameters: I/O Port Read dialog box opens.

**2** In the **I/O-Port address** box, enter the beginning address for each value this block reads.

For example, if you want to read a word (16 bits) starting at I/O port 0x300, followed by a byte (8 bits) at 0x302, enter

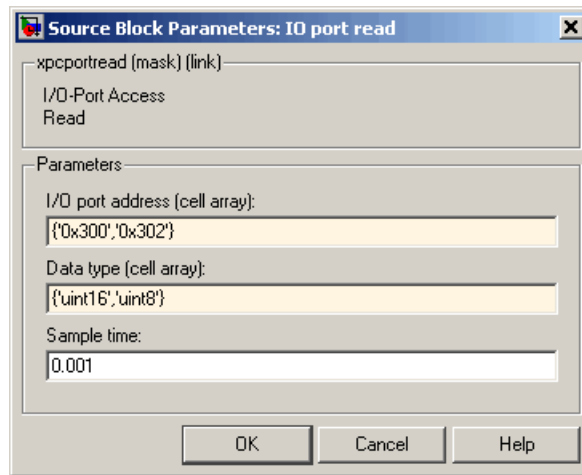
```
{ '0x300', '0x302' }
```

**3** In the **Data type** box, enter the type for each value this block reads. There is one type for each address you entered in the **I/O-Port address** box.

For example if you want to read a word and then a byte, enter

```
{ 'uint16', 'uint8' }
```

Your dialog box should look similar to the figure shown below.



#### 4 Click **OK**.

The number of outputs from the block changes to reflect the length of the I/O-Port address cell array.

## Block Parameters

### **I/O-Port address (cell array)**

This is the cell array containing the beginning I/O port addresses for the data you want to read. These addresses are specified in terms of hexadecimal strings.

### **Data type (cell array)**

This is the cell array containing the types of data you want to read from I/O port. The **Data type** cell array has one value for each value in the **I/O-Port address** cell array.

The type `uint32` reads a double word (32 bits), `uint16` reads a word, and a `uint8` reads a byte.

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

# I/O Port Write

---

<b>Purpose</b>	I/O Port Write
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	Intel 80x86 and compatible processors have a special address space reserved for I/O devices.

**Example** To access this special address space and communicate directly to a device, xPC Target provides the I/O Port Read and I/O Port Write blocks. These blocks enable the transfer of data from and to the I/O ports (see also, I/O Port Read).

**1** Double click the I/O Port Write block.

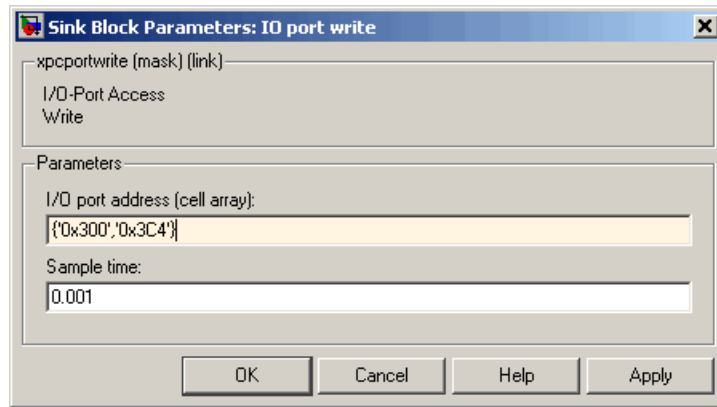
The Block Parameters: I/O Port Write dialog box opens.

**2** Enter the parameters.

For example, if you want to write a double word (32 bits) starting at I/O port 0x300, followed by a word (16 bits) at 0x304, enter

```
{ '0x300' , '0x304' }
```

Your dialog box should look similar to the figure shown below.



### 3 Click **OK**.

The number of inputs to the block changes to reflect the length of the I/O-Port address cell array. The data type of the input signal reflects the type of value written to the I/O port. For example, an input signal of type `uint32` writes a double word, an `uint16` input signal writes a word, and an `uint8` input signal writes a byte.

## Block Parameters

### I/O-Port address (cell array)

This is the cell array containing the I/O port addresses for the data that you want read. These addresses are specified in terms of hexadecimal strings.

### Sample time

Enter a base sample time or a multiple of the base sample time.

# Pentium Time Stamp Counter

---

**Purpose** Pentium time stamp counter

**Library** xPC Target Library for Misc.

**Description** This block outputs a uint32 vector of length 2 that contains the time stamp counter value (rdtsc).

---

**Note** Use this block for Pentium machines only. Using this block on non-Pentium machines might result in abnormal behavior, including crashes. This block takes an arbitrary input used only to sequence execution of the block.

---

**Block Parameters** None

**Purpose** Time stamp delta

**Library** xPC Target Library for Misc.

**Description** This block takes as input two uint32 vectors of length 2. Two separate Pentium Time Stamp Counter blocks can supply each vector. The output is a scalar double that contains the delta between the two time stamps.

**Block Parameters** None

# Scope (xPC) (xPC Target Scope)

---

**Purpose** Scope (xPC) (xPC Target Scope block)

**Library** xPC Target Library for Misc.

**Description** For information about this block, see “Adding an xPC Target Scope Block” in this document and “Entering Parameters for an xPC Target Scope Block” of Getting Started with xPC Target.

There are three types of scopes, target, host, and file. The xPC Target Scope block dialog changes depending on which scope type (**Scope type**) you select. By default, the scope dialog displays the scope of type Target dialog. The following sections describe the parameters depending on the scope type:

- “Scope of Type Target Block Parameters” on page 39-24
- “Scope of Type Host Block Parameters” on page 39-28
- “Scope of Type File Block Parameters” on page 39-30

## Scope of Type Target Block Parameters

### Scope number

Contains a unique number to identify the scope that is displayed. This number is incremented each time you add a new xPC Target Scope block. Normally, you do not want to edit this value.

This number identifies the xPC Target Scope block and the scope screen on the host or target computer.

### Scope type

From the list, select Target if it is not already selected.

The updated dialog box displays.

### Start scope when application starts

Select this check box to start a scope when the target application is downloaded and started. The scope window opens automatically.



## Scope mode

From the list, select either Numerical, Graphical redraw, Graphical sliding, or Graphical rolling.

If you have a scope type of Target and a scope mode of Numerical, the scope block dialog adds a **Numerical format** box to the dialog. You can define the display format for the data. If you choose not to complete the **Numerical format** box, xPC Target displays the signal using the default format of %15.6f, which is a floating point format, with no label.

## Numerical format

If you have a scope type of Target and a scope mode of Numerical, the scope block dialog adds a **Numerical format** box to the dialog. Enter a label and associated numeric format type in which to display signals. By default, the entry format is the floating point %15.6f. The **Numerical format** box takes entries of the format:

```
'[LabelN] [%width.precisiontype] [LabelX]'
```

where

LabelN is the label for the signal. You can use a different label for each signal or the same label for each signal. This argument is optional.

width is the minimum number of characters to offset from the left of the screen or label. This argument is optional.

precision is the maximum number of decimal points for the signal value. This argument is optional.

type is the data type for the signal format. You can use one or more of the following types:

## Scope (xPC) (xPC Target Scope)

---

Type	Description
%e or %E	Exponential format using e or E
%f	Floating point
%g	Signed value printed in f or e format depending on which is smaller
%G	Signed value printed in f or E format depending on which is smaller

LabelX is a second label for the signal. You can use a different label for each signal or the same label for each signal. This argument is optional.

Enclose the contents of the **Numerical format** field in quotation marks.

For example

```
'Foo %15.2f end'
```

For a whole integer signal value, enter 0 for the precision value. For example

```
'Foo1 %15.0f end'
```

For a multiple format entries, delimit each entry with a comma and surround the entire string with a pair of quotes. For example

```
'Foo2 %15.6f end,Foo3 %15.6f end2'
```

You can have multiple **Numerical format** entries, separated by a comma. If you enter one entry, that entry applies to each signal (scalar expansion). If you enter fewer label entries than signals, the first entry applies to the first signal, the second entry applies to the second signal, and so forth, and the last entry is scalar expanded for the remaining signals. If you have two entries and one signal, xPC Target ignores the second label entry and applies

the first entry. You can enter as many format entries as you have signals for the scope.

### **Grid**

Select this check box to display grid lines on the scope. Note that this parameter is only applicable for scopes of type Target and scope modes of type Graphical redraw, Graphical sliding, and Graphical rolling.

### **Y-Axis limits**

Enter a row vector with two elements where the first element is the lower limit of the  $y$ -axis and the second element is the upper limit. If you enter 0 for both elements, then the scaling is set to auto. Note that this parameter is only applicable for scopes of type Target and scope modes of type Graphical redraw, Graphical sliding, and Graphical rolling.

### **Number of samples**

Enter the number of values to be acquired in a data package.

If you select a **Scope mode** of Graphical redraw, this parameter specifies the number of values to be acquired before the graph is redrawn.

If you select a **Trigger mode** other than FreeRun, this parameter can specify the number of samples to be acquired before the next trigger event.

### **Number of pre/post samples**

Enter the number of samples to save or skip. Specify a value less than 0 to save this number of samples before a trigger event. Specify a value greater than 0 to skip this number of samples after the trigger event before data acquisition begins.

### **Decimation**

Enter a value to collect data at each sample time (1) or to collect data at less than every sample time (2 or greater).

# Scope (xPC) (xPC Target Scope)

---

## Trigger mode

From the list, select either FreeRun, Software Triggering, Signal Triggering, or Scope Triggering.

If you select FreeRun or Software Triggering, you do not need to specify anything else.

If you select Signal Triggering, then in the **Trigger signal** box, enter the index of a signal. In the **Trigger level** box, enter a value for the signal to cross before triggering. From the **Trigger slope** list, select either, rising, or falling. You do not need to specify the **Trigger scope number**.

If you select Scope Triggering, then in the **Trigger scope number** box, enter the scope number of a Scope block. If you use this trigger mode, you must also add a second Scope block to your Simulink model. You do not need to specify the **Trigger signal**.

If you want the scope to trigger on a specific sample of the other scope, enter a value in the **Sample to trigger on** box. The default value is 0 and indicates that the triggering scope and the triggered (current) scope start simultaneously. For more information on this field, see “Triggering One Scope with Another Scope to Acquire Data” in xPC Target User’s Guide.

For additional information on trigger, see “Advanced Data Acquisition Topics”.

## Scope of Type Host Block Parameters

### Scope number

Contains a unique number to identify the scope that is displayed. This number is incremented each time you add a new xPC Target scope block. Normally, you do not want to edit this value.

This number identifies the xPC Target Scope block and the scope screen on the host or target computer.

## Scope type

From the list, select Host.

The updated dialog box is displayed.

## Start scope when application starts

Select this check box to start a scope when the target application is downloaded and started. With a scope of type target, the scope window opens automatically. With a scope of type host, you can open a host scope viewer window from xPC Target Explorer.

## Number of samples

Enter the number of values to be acquired in a data package.

## Number of pre/post samples

Enter the number of samples to save or skip. Specify a value less than 0 to save this number of samples before a trigger event. Specify a value greater than 0 to skip this number of samples after the trigger event before data acquisition begins.

## Decimation

Enter a value to collect data at each sample time (1) or to collect data at less than every sample time (2 or greater).

## Trigger Mode

From the list, select either FreeRun, Software Triggering, Signal Triggering, or Scope Triggering.

If you select FreeRun or Software Triggering, the trigger event is an automatic one. No other trigger specification is required (see trigger for further details on software trigger).

If you select Signal Triggering, then in the **Trigger signal** box, enter the index of a signal. In the **Trigger level** box, enter a value for the signal to cross before triggering. From the **Trigger slope** list, select either, rising, or falling. You do not need to specify scope triggering.

# Scope (xPC) (xPC Target Scope)

---

If you select Scope Triggering, then in the **Trigger scope number** box, enter the scope number of a Scope block. If you use this trigger mode, you must also add a second Scope block to your Simulink model. You do not need to specify signal triggering.

If you want the scope to trigger on a specific sample of the other scope, enter a value in the **Sample to trigger on** box. The default value is 0 and indicates that the triggering scope and the triggered (current) scope start simultaneously. For more information on this field, see “Triggering One Scope with Another Scope to Acquire Data” in the xPC Target User’s Guide.

## Scope of Type File Block Parameters

### Scope number

Contains a unique number to identify the scope that is displayed. This number increments each time you add a new xPC Target scope block. Normally, you do not want to edit this value.

This number identifies the xPC Target Scope block and the scope screen on the host or target computer.

### Scope type

From the list, select File.

The updated dialog box is displayed.

### Start scope when application starts

Select the check box to start a scope when the target application is downloaded and started. The scope window opens automatically.

### Number of samples

Enter the number of values to be acquired in a data package. This parameter works in conjunction with the **AutoRestart** check box. If the **AutoRestart** box is selected, the scope of type file collects data up to **Number of samples**, then starts over again, overwriting the buffer. If the **AutoRestart** box is not selected, the scope of type file collects data only up to **Number of samples**, then stops.

## Number of pre/post samples

Enter the number of samples to save or skip. Specify a value less than 0 to save this number of samples before a trigger event. Specify a value greater than 0 to skip this number of samples after the trigger event before data acquisition begins.

## Decimation

Enter a value to collect data at each sample time (1) or to collect data at less than every sample time (2 or greater).

## Trigger Mode

From the list, select either FreeRun, Software Triggering, Signal Triggering, or Scope Triggering.

If you select FreeRun or Software Triggering, the trigger event is an automatic one. No external trigger specification is required.

If you select Signal Triggering, then in the **Trigger signal** box, enter the index of a signal. In the **Trigger level** box, enter a value for the signal to cross before triggering. From the **Trigger slope** list, select either, rising, or falling. You do not need to specify scope triggering.

If you select Scope Triggering, then in the **Trigger scope number** box, enter the scope number of a Scope block. If you use this trigger mode, you must also add a second Scope block to your Simulink model. You do not need to specify signal triggering.

If you want the scope to trigger on a specific sample of the other scope, enter a value in the **Sample to trigger on** box. The default value is 0 and indicates that the triggering scope and the triggered (current) scope start simultaneously. For more information on this field, see “Triggering One Scope with Another Scope to Acquire Data” in the xPC Target User’s Guide.

## Filename

Enter a name for the file to contain the signal data. By default, the target PC writes the signal data to a file named C:\data.dat.

# Scope (xPC) (xPC Target Scope)

---

## Mode

From the list, select either Lazy or Commit. Both modes open a file, write signal data to the file, then close that file at the end of the session. With the Commit mode, each file write operation simultaneously updates the FAT entry for the file. This mode is slower, but the file system always knows the actual file size. With the Lazy mode, the FAT entry is updated only when the file is closed and not during each file write operation. This mode is faster, but if the system crashes before the file is closed, the file system might not know the actual file size (the file contents, however, will be intact).

## WriteSize

Enter the block size, in bytes, of the data chunks. This parameter specifies that a memory buffer, of length **Number of samples**, writes data to the file in **WriteSize** chunks. By default, this parameter is 512 bytes, which is the typical disk sector size. Using a block size that is the same as the disk sector size provides optimal performance.

## Number of samples

Enter the number of values to be acquired in a data package.

## AutoRestart

Select this check box to enable the scope of type file to collect data up to **Number of samples**, then start over again, appending the new data to the end of the signal data file. Clear the **AutoRestart** check box to have the scope of type file collect data up to **Number of samples**, then stop.



**Purpose**

Software Reboot

**Library**

xPC Target Library for Misc.

**Description**

You can use the Software Reboot block to reboot the target PC when your simulation reaches a certain state. For example, if your control system becomes unstable you may want to reboot the target PC.

This block has one input. The input is the reboot signal and accepts the following values:

- 1 — Reboots the target PC.
- 0 — If you use a Software Reboot block and you don't want the target PC to reboot, the input value to this block must be 0.

---

**Note** Not all systems support rebooting from software. You can test whether your system supports a software reboot. From the MATLAB Command Window, enter `xpctest`. For more information, see “Testing and Troubleshooting the Installation”.

---

**Block****Parameters****Sample time**

Enter a base sample time or a multiple of the base sample time.

# To xPC Target

---

**Purpose** Data to target PC

**Description** This block behaves as a sink. The main purpose of this block is to write a new value to a specific parameter on the target application. This block is implemented as an M-file S-function. The block is optimized so that it only changes a parameter on the target application when the input value differs from the value that existed at the last time step. This block uses the parameter downloading feature of the xPC Target command-line interface.

---

**Note** The use of To xPC Target blocks requires a connection between the host and target PC. If there is no connection between the host and target PC, operations such as opening a model that contains these blocks or copying these blocks within or between models, will take significantly longer than normal.

---

Some notes on the To xPC Target block behavior:

- To highlight the Simulink model block referenced by a To xPC Target block, double-click the To xPC Target block.
- If the To xPC Target block has not yet been configured, double-clicking the To xPC Target block has no effect.
- To edit the To xPC Target block parameters, right-click on the block and select **Mask Parameters**.

## Block Parameters

### **xPC Target application name**

The function `xpcsliface` automatically enters a name entry for this parameter. It is the same name as the Simulink model that xPC Target uses to build the target application.

### **Path to block in target application**

The function `xpcsliface` automatically enters an entry for this parameter and uses it to access the block identifier.

### **Parameter name**

The function `xpcsliface` automatically determines the entry for this parameter and enters it. Note that the parameter name might not match the label name for that parameter in the Block Parameters dialog box. For example, the label name for a gain block is `Constant value`, but the parameter name is **Value**.

### **Use default target PC**

Selecting this option directs the To xPC Target block to communicate to the default target PC. This assumes that you configured a default target PC through the xPC Target Explorer (see sections “Serial Communication” and “Network Communication” if you have not). By default, this check box is selected.

### **Specify target name**

If you deselect the **Use default target PC** check box, this field is displayed. Enter the name of the configured target PC.

### **See Also**

“Creating a Custom Graphical Interface” in the xPC Target User’s Guide

# xPC Target TET

---

<b>Purpose</b>	xPC Target TET
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	<p>This block outputs the Task Execution Time (TET) in seconds.</p> <p>Use the output of this block as an input to an xPC Target Scope block. This allows you to visualize the TET while your target application is running.</p>
<b>Block Parameters</b>	<p><b>Sample time</b> Enter a base sample time or a multiple of the base sample time.</p>

<b>Purpose</b>	xPC Target Time
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	<p>This block outputs the time in clock ticks that the kernel has been executing. This is not the same as the execution time of the target application, but it is the time since you booted up the target PC.</p> <p>A use for this block is to determine the execution time of subsystems in your Simulink model. Add an xPC Target Time block before and after the subsystem, and then calculate the difference in time.</p> <p>The time is in clock ticks, and it uses the target PC interval timer to generate the ticks. Since the interval timer runs at a frequency of 1.193 Mhz, you can calculate the time in seconds by dividing the output by <math>1.193 \times 10^6</math>.</p>
<b>Block Parameters</b>	<p><b>Sample time</b> Enter a base sample time or a multiple of the base sample time.</p>



# Utility Blocks

---

The xPC Target library contains utility blocks that support a number of functions. One particular sublibrary does not contain these utility blocks. Note the **Library** heading for the location of the block.

## **Blocks – Alphabetical List**



<b>Purpose</b>	Construct data frames
<b>Library</b>	xPC Target Library for Utilities
<b>Description</b>	This block constructs data frames. Its output port is typically connected to an input port of a Send block or Digital Output block. The block has one output port. This port can be a vector of any size; it represents the data frame entity constructed by the signals entering the block at its input ports. The number of input ports depends on the setting in the block's dialog box.
<b>Block Parameters</b>	<p><b>Bit Patterns</b></p> <p>Specify bit patterns. The data type entered in the control must be a MATLAB cell array vector. The number of elements in the cell array define the number of input ports shown by this block instance. The cell array elements must be of type double array and define the position of each bit of the incoming value (data typed input port) in the outgoing double value (data frame). From a data type perspective (input ports), the block behaves like a Simulink Sink block, and therefore the data types of the input ports are inherited from the driving blocks.</p> <p><b>Output port (packed) data type</b></p> <p>From the list, select an output port (packed) data type.</p> <ul style="list-style-type: none"><li>• double</li><li>• single</li><li>• int8</li><li>• uint8</li><li>• int16</li><li>• uint16</li><li>• int32</li><li>• uint32</li><li>• boolean</li></ul>

# Bit Packing

---

## **Output port (packed) dimensions**

Specify the dimensions the output port (packed). Enter this as a vector. Specify the size of the port using a format compatible with the MATLAB `size` command.

<b>Purpose</b>	Deconstruct data frames
<b>Library</b>	xPC Target Library for Utilities
<b>Description</b>	<p>This block is used to extract data frames. Its input port is typically connected to an output port of a Receive block or Digital Input block.</p> <p>The block has one input port, which represents the data frame entity from which the signals are extracted and leaving the block at its output ports. The number of output ports and the data type of each output port depend on the settings in the block's dialog box.</p>
<b>Block Parameters</b>	<p><b>Bit Patterns</b></p> <p>Specify bit patterns. The data type must be a MATLAB cell array vector. The number of elements in the cell array define the number of input ports shown by this block instance. The cell array elements must be of type double array and define the position of each bit of the incoming value (data typed input port) in the outgoing double value (data frame). From a data type perspective, the block behaves like a Sink block. The <b>Input port (packed) data types</b> specify the data type of the input port.</p> <p><b>Input port (packed) data types</b></p> <p>From the list, select an input port (packed) data type.</p> <ul style="list-style-type: none"><li>• double</li><li>• single</li><li>• int8</li><li>• uint8</li><li>• int16</li><li>• uint16</li><li>• int32</li><li>• uint32</li><li>• boolean</li></ul>

# Bit Unpacking

---

## **Input port (packed) dimension**

Specify the dimensions of the input port (packed). Enter this as a vector. Specify the size of the port using a format compatible with the MATLAB `size` command.

## **Output port (unpacked) data types (cell array)**

The output ports (packed) can be any data type. The number of elements in the cell array define the number of output ports shown by this block instance. The data types can be

- `double`
- `single`
- `int8`
- `uint8`
- `int16`
- `uint16`
- `int32`
- `uint32`
- `boolean`

## **Output port (unpacked) dimension (cell array)**

Specify the dimensions of each output port (unpacked). Enter this as a cell array of vector sizes.

## **Sign extend**

Select this check box to enable sign extension. If you select this check box and unpack the data frame into a signed type (`int8`, `int16`, or `int32`), the block performs sign extension as necessary. For example, if the bit pattern is `[0:4]`, and the data type is `int8`, you are extracting 5 bits into an 8-bit wide signed type. In this case, bits 5, 6, and 7 are the same as bit 4, resulting in the proper sign extension. This functionality enables you to pack and unpack negative numbers without losing accuracy.

<b>Purpose</b>	Construct data frames
<b>Library</b>	xPC Target Library for Utilities
<b>Description</b>	The Byte Packing block converts one or more Simulink signals of user-selectable data types to a single vector of varying data types. The output of this block is typically connected to an input port of a Send block.
<b>Block Parameters</b>	<p><b>Output port (packed) data type</b> Specify the data type for the different signals. The block determines the sizes of the signals automatically. From the list, select an output port (packed) data type.</p> <ul style="list-style-type: none"><li>• double</li><li>• single</li><li>• int8</li><li>• uint8</li><li>• int16</li><li>• uint16</li><li>• int32</li><li>• uint32</li><li>• boolean</li></ul> <p><b>Inport port (unpacked) data types (cell array)</b> Specify the data types of the input ports (unpacked) for the different signals as part of the block parameters.</p> <ul style="list-style-type: none"><li>• double</li><li>• single</li><li>• int8</li><li>• uint8</li></ul>

# Byte Packing

---

- int16
- uint16
- int32
- uint32
- boolean

The data type must be a MATLAB cell array vector. The number of elements in the cell array define the number of input ports shown by this block instance.

## **Byte Alignment**

The byte alignment field specifies how the data types are aligned. The possible values are 1, 2, 4, and 8. The byte alignment scheme is simple, and ensures that each element in the list of signals starts on a boundary specified by the alignment relative to the start of the vector.

<b>Purpose</b>	Deconstruct data frames
<b>Library</b>	xPC Target Library for Utilities
<b>Description</b>	<p>This block is the exact analog of the Pack block. It can receive a vector of one of the following types:</p> <ul style="list-style-type: none"><li>• double</li><li>• single</li><li>• int8</li><li>• uint8</li><li>• int16</li><li>• uint16</li><li>• int32</li><li>• uint32</li><li>• boolean</li></ul> <p>It outputs various Simulink data types in different sizes.</p>
<b>Block Parameters</b>	<p><b>Output port (unpacked) data types (cell array)</b></p> <p>The output ports (unpacked) can be any data type. The number of elements in the cell array define the number of output ports shown by this block instance. The data types can be</p> <ul style="list-style-type: none"><li>• double</li><li>• single</li><li>• int8</li><li>• uint8</li><li>• int16</li><li>• uint16</li></ul>

# Byte Unpacking

---

- int32
- uint32
- boolean

## **Output port (unpacked) dimensions (cell array)**

Specify the dimensions of each output port (unpacked). Enter this as a cell array of vector sizes.

## **Byte Alignment**

The byte alignment field specifies how the data types are aligned. The possible values are 1, 2, 4, and 8. The byte alignment scheme is simple, and ensures that each element in the list of signals starts on a boundary specified by the alignment relative to the start of the vector.



<b>Purpose</b>	Construct data frames
<b>Library</b>	xPC Target Library for Digital I/O
<b>Description</b>	<p>This block constructs data frames. Its output port is typically connected to an input port of a digital I/O block. The block has one output port of data type uint32 (a scalar), which represents the data frame entity constructed by the signals entering the block at its input ports. The number of input ports depends on the setting in the block's dialog box. See CAN Bit-Packing for an example of how to use a bit-packing block.</p>
<b>Block Parameters</b>	<p><b>Bit Patterns</b></p> <p>Specify bit patterns. The data type entered in the control must be a MATLAB cell array vector. The number of elements in the cell array define the number of input ports shown by this block instance. The cell array elements must be of type double array and define the position of each bit of the incoming value (data typed input port) in the outgoing uint32 value (data frame).</p> <p>From a data type perspective (input ports), the block behaves like a Simulink Sink block. Therefore, the data types of the input ports are inherited from the driving blocks.</p> <p>This block inherits the sample time of the driving blocks.</p>
<b>See Also</b>	Digital I/O Bit-Unpacking

# Digital I/O Bit-Unpacking

---

**Purpose** Extract data frames

**Library** xPC Target Library for Digital I/O

**Description** This block is used to extract data frames. Its input port is typically connected to an output port of a digital I/O block. The block has one input port of data type `uint32` (a scalar), which represents the data frame entity from which the signals are extracted and leaving the block at its output ports. The number of output ports and the data type of each output port depend on the settings in the block's dialog box. See CAN Bit-Unpacking for an example of how to use a bit-unpacking block.

## Block Parameters

### Bit Patterns

Lets you define the bit patterns in a flexible way. The data type entered in the field must be a MATLAB cell array vector. The number of elements in the cell array define the number of output ports shown by this block instance. The cell array elements must be of type double array and define the position of each bit of the incoming value (data typed output port) in the incoming `uint32` value (data frame).

### Sign extend

Select this check box to enable sign extension. If you select this check box and unpack the data frame into a signed type (`int8`, `int16`, or `int32`), the block performs sign extension as necessary. For example, if the bit pattern is `[0:4]`, and the data type is `int8`, you are extracting 5 bits into an 8-bit wide signed type. In this case, bits 5, 6, and 7 are the same as bit 4, resulting in the proper sign extension. This functionality enables you to pack and unpack negative numbers without losing accuracy. In the example from CAN Bit-Unpacking, you can pack and unpack numbers in the range `[-16 : 15]` (a fictitious `int5` type).

### Data Types

From a data type perspective (output ports), the block behaves like a Simulink Source block. Therefore, the data types of the output ports must be defined in this field. The data type entered

must be a MATLAB cell array vector of the same length as the bit pattern cell array. The cell array elements must be of type char and define the data type of the corresponding output port. The following values are supported:

- boolean
- int8
- uint8
- int16
- uint16
- int32
- uint32

This block inherits the sample time of the driving blocks.

### **See Also**

Digital I/O Bit-Packing

# Extended Counter

---

<b>Purpose</b>	Sample extended counter
<b>Library</b>	xPC Target Library for Incremental Encoder/Utilities
<b>Description</b>	This block outputs a count that is not bound by the number of bits on the board encoder counter register. It receives as input the output of an encoder block.
<b>Block Parameters</b>	<p><b>Rollover Counts</b> Enter the number of rollover counts. This is the number at which the rollover occurs. This value should normally be <math>2^n</math>, where <math>n</math> is the number of bits in the encoder counter register. Each rollover equals this number.</p> <p><b>Encoder Starting Value</b> Enter the starting value with which the encoder block begins. This value should match the value in the encoder mask.</p> <p><b>Rollover Threshold</b> Enter a rollover threshold value. This subsystem infers when a rollover occurs by comparing successive outputs from an encoder block. When the difference between outputs is extremely large, a rollover has occurred. Enter a value that is</p> <ul style="list-style-type: none"><li>• Larger than the worst-case difference between successive outputs (to prevent declaring a rollover when it was just a large step)</li><li>• Less than the maximum count value minus the worst-case difference (to prevent missing a rollover)</li></ul> <p>Ideally, enter the value <math>2^{(n-1)}</math>, where <math>n</math> is the number of bits in the encoder counter register. The default is <math>2^{15}</math>.</p>
<b>See Also</b>	Rollover Counter

<b>Purpose</b>	Sample rollover counter
<b>Library</b>	xPC Target Library for Incremental Encoder/Utilities
<b>Description</b>	This block counts the number of times the output of an encoder block has rolled over. It counts up for positive direction rollovers and down for negative direction rollovers.
<b>Block Parameters</b>	<p><b>Encoder Starting Value</b> Enter the starting value you want the encoder block to begin with. Match this value to the value in the encoder mask.</p> <p><b>Rollover Threshold</b> Enter a rollover threshold value. To determine when a rollover has occurred, the model compares two consecutive encoder outputs and looks for a large jump. A large jump indicates a register overflow.</p> <p>The threshold you enter defines a large jump. The smaller the threshold, the more likely the model will misinterpret a jump in successive encoder counts as a rollover. The larger the threshold, the more likely the model will not detect a rollover. Ideally, choose a number that is half the size of the register (for example, <math>2^{(n-1)}</math>), where <math>n</math> is the size of the register in bits.</p>
<b>See Also</b>	Extended Counter

# Shared Memory Pack

---

<b>Purpose</b>	Shared memory pack
<b>Library</b>	xPC Target Library for Shared Memory
<b>Description</b>	<p>This block packs the specified partition structure into an unstructured double word array vector by converting one or more Simulink signals of varying data types into the vector. Typically, the input to a pack block is the output from a write block. Simulink is not aware of structures; you must pass the output of each structure segment as input to the Shared Memory Pack block.</p> <p>Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. See the following, depending on the shared memory manufacturer:</p> <ul style="list-style-type: none"><li>• “SBS Shared Memory Structure Reference” on page 31-8</li><li>• “SBS Shared Memory Structure Reference” on page 31-8</li><li>• “VMIC Shared Memory Structure Reference” on page 38-6</li></ul> <p>Memory partitions consist of groups of Simulink signals, which are combined into blocks (packets) of 32-bit words.</p> <p>This block ignores the <b>Address</b> field of the partition structure.</p>
<b>Block Parameters</b>	<p><b>Partition struct</b> Enter the name of the predefined shared memory partition structure.</p>
<b>See Also</b>	<p>Shared Memory Unpack</p> <p>“SBS Shared Memory Structure Reference” on page 31-8</p> <p>“SBS Shared Memory Structure Reference” on page 31-8</p> <p>“VMIC Shared Memory Structure Reference” on page 38-6</p>

<b>Purpose</b>	Shared memory unpack
<b>Library</b>	xPC Target Library for Shared Memory
<b>Description</b>	<p>This block unpacks an unstructured double word array vector (from the Shared Memory Pack block) into the specified partition structure.</p> <p>Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. See the following, depending on the shared memory manufacturer:</p> <ul style="list-style-type: none"><li>• “SBS Shared Memory Structure Reference” on page 31-8</li><li>• “Systran Shared Memory Structure Reference” on page 34-6</li><li>• “VMIC Shared Memory Structure Reference” on page 38-6</li></ul> <p>This block ignores the <b>Address</b> field of the partition structure.</p>
<b>Block Parameters</b>	<p><b>Partition struct</b></p> <p>Enter the name of the predefined shared memory partition structure. The block unpacks the double word array vector into this structure.</p>
<b>See Also</b>	<p>Shared Memory Pack</p> <p>“SBS Shared Memory Structure Reference” on page 31-8</p> <p>“SBS Shared Memory Structure Reference” on page 31-8</p> <p>“VMIC Shared Memory Structure Reference” on page 38-6</p>





# Obsolete Drivers

---

The following drivers are obsolete in xPC Target:

xPC Target Library of Obsolete Drivers (p. 41-2)      Library of obsolete I/O drivers

Blocks — Alphabetical List (p. 41-3)

## xPC Target Library of Obsolete Drivers

You can still access obsolete driver blocks by typing the following command at the MATLAB Command Window:

```
xpcobsoletelib
```

This library contains obsolete drivers for the following manufacturers:

- Burr\_brown
- Gespac

---

**Note** Existing models that use these drivers will still work. However, their presence is not guaranteed for future releases. Do not use these driver blocks in new models.

---

In addition, in the xPC Target library (`xpc1ib`), there are new versions of the following Measurement Computing Incremental Encoder drivers. Please use the new version of these drivers.

- CIO-QUAD02
- CIO-QUAD04
- PCI-QUAD04

To access the previous versions of these drivers, use the `xpcobsoletelib` command. The presence of the obsolete versions of these drivers is not guaranteed for future releases.

## **Blocks – Alphabetical List**

# Burr-Brown PCI-20003M

---

**Burr-Brown** This section describes the Burr-Brown boards supported by xPC Target (<http://www.instrument.com/pci>).

**Board** Burr-Brown PCI-20003M

**General Description** The PCI-20003M is an I/O board with two analog output (D/A) channels (12-bit).

xPC Target supports this board when it is installed on a PCI-20041C carrier board with this driver block:

- Burr-Brown PCI-20003M Analog Output (D/A)

<b>Board Characteristics</b>	Board name	PCI-20003M
	Manufacturer	Burr-Brown
	Bus type	ISA
	Access method	Memory mapped
	Multiple block instance support	No
	Multiple board support	Yes

# Burr-Brown PCI-20003M Analog Output (D/A)

**Purpose** PCI-20003M Analog Output block

**Library** xPC Target Library for Burr-Brown

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

## Block Parameters

### Channel vector

This parameter is a combined Channel vector and Range vector. The number of elements defines the number of A/D channels used.

Enter a range code for each of the A/D channels used. This driver allows a different range for each channel with a maximum of two A/D channels.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 to +10 volts and the second channel is 0 to +5 volts, enter

[ -10,5]

The following jumpers on the module must correspond to this range setting.

- W1 to W5, W13, W14, W27, and W31

# Burr-Brown PCI-20003M Analog Output (D/A)

---

- W7 to W11, W30, and W32

## **Sample Time**

Enter the base sample time or a multiple of the base sample time.

## **Module Number**

Enter a number from 1 to 3 to identify the connector on the carrier board where the I/O module is inserted. This driver verifies if the module is placed on the specified module connector.

## **Base Address or Carrier Board (ie: 0xd000)**

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

**Board** Burr-Brown PCI-20019M

**General Description**

The PCI-20019M is an I/O board with 8 single analog input (A/D) channels (12-bit).

xPC Target supports this board when it is installed on a PCI-20041C carrier board with this driver block:

- Burr-Brown PCI-20019M Analog Input (A/D)

**Board Characteristics**

Board name	PCI-20019M
Manufacturer	Burr-Brown
Bus type	ISA
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# Burr-Brown PCI-20019M Analog Input (A/D)

---

**Purpose** PCI-20019M Analog Input block

**Library** xPC Target Library for Burr-Brown

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

## Block Parameters

### Number of Channels

Enter a number between 1 and 8 to select the number of A/D channels used. This driver does not allow the selection of individual channels.

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

### Input Range

Enter an input range code for all A/D channels. This driver does not allow the selection of a different range for individual channel. The input range is the same for all A/D channels

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-2.5 to +2.5	-2.5
0 to 10	10
0 to 5	5



# Burr-Brown PCI-20019M Analog Input (A/D)

The jumpers W1 to W5 on the module must correspond to this range setting.

## Sample time

Enter a base sample time or a multiple of the base sample time.

## Module Number (1-3)

Enter a number from 1 to 3 to identify the connector on the carrier board where the I/O module is inserted. This driver verifies if the module is placed on the specified module connector.

## Base Address of Carrier Board (ie. 0xd000)

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

Other jumper on this board. The switch and jumper settings, that are not mentioned here, have no influence on the running of xPC Target.

Jumper Number	Jumper
W6	out
W8	in
W10	out
W11	in
W12	out
W22	out
W27	out
W30	-
W31	-

# Burr-Brown PCI-20023M

---

**Board** Burr-Brown PCI-20023M

**General Description**

The PCI-20023M is an I/O board with 8 single analog input (A/D) channels (12-bit).

xPC Target supports this board when it is installed on a PCI-20041C carrier board with this driver block:

- Burr-Brown PCI-20023M Analog Input (A/D)

**Board Characteristics**

Board name	PCI-20023M
Manufacturer	Burr-Brown
Bus type	ISA
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# Burr-Brown PCI-20023M Analog Input (A/D)

**Purpose** PCI-20023M Analog Input block

**Library** xPC Target Library for Burr-Brown

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

## Block Parameters

### Number of Channels

Enter a number between 1 and 8 to select the number of A/D channels used. This driver does not the selection of individual channels.

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

### Input Range

Enter an input range code for all A/D channels. This driver does not allow the selection of a different range for individual channel. The input range is the same for all A/D channels.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

The jumpers W1, W2, W4, W5, W33 on the module must correspond to this range setting. The switch and jumper settings, that are not mentioned here, have no influence on running xPC Target.

# Burr-Brown PCI-20023M Analog Input (A/D)

---

## Sample time

Enter the base sample time or a multiple of the base sample time.

## Module Number (1-3)

Enter a number from 1 to 3 to identify the connector on the carrier board where the I/O module is inserted. This driver verifies if the module is placed on the specified module connector.

## Base Address of Carrier Board (ie. 0xd000)

Enter the base address of the I/O board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

Other jumpers on this board. The switch and jumper settings, that are not mentioned here, have no influence on the running of xPC Target.

Jumper Number	Jumper
W6	out
W8	in
W9	-
W10	out
W11	in
W12	out
W27	out
W30	-
W31	-

**Board** Burr-Brown PCI-20041C

**General Description** The PCI-20041C is a carrier board with 32 digital I/O-lines grouped into four ports that can be configured as digital input or output. Each port has a maximum of 8 digital lines.

xPC Target supports this board with these driver blocks:

- Burr-Brown PCI-20041C Digital Input
- Burr-Brown PCI-20041C Digital Output

<b>Board Characteristics</b>	Board name	PCI-20041C
	Manufacturer	Burr-Brown
	Bus type	ISA
	Access method	Memory mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Burr-Brown PCI-20041C Digital Input

---

**Purpose** PCI-20041C Digital Input block

**Library** xPC Target Library for Burr-Brown

**Note** Use a separate driver block for each port. By selecting the digital input driver block, the port is configured as input.

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

## **Block Parameters**

### **Number of Channels**

Enter a number between 1 and 8 to select the number of digital input lines used with this port.

### **Port Number (1-4)**

Enter a number from 1 to 4 to identify the port used with this block of digital input lines.

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

### **Module Number (0-3)**

Enter a number from 0 to 3 to identify the connector on the carrier board where the I/O module is inserted. This driver verifies if the module is placed on the specified module connector.

### **Base Address or Carrier Board (ie: 0xd000)**

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

# Burr-Brown PCI-20041C Digital Output

**Purpose** PCI-20041C Digital Output block

**Library** xPC Target Library for Burr-Brown

**Note** Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### Number of Channels

Enter a number between 1 and 8 to select the number of digital output lines used with this port.

### Port Number (0-3)

Enter a number from 0 to 3 to identify the port used with this block of digital output lines.

### Sample Time

Enter a base sample time or a multiple of the base sample time.

### Module Number (1-3)

Enter a number from 1 to 3 to identify the connector on the carrier board where the I/O module is inserted. This driver verifies if the module is placed on the specified module connector.

### Base Address or Carrier Board (ie: 0xd000)

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 d000 (hexadecimal), enter

0xd000

# Burr-Brown PCI-20098C

---

**Board** Burr-Brown PCI-20098C

**General Description** The PCI-20098C is a carrier board with 8 single or 16 differential analog input (A/D) channels (12-bit), and 16 digital I/O-lines grouped into two 8-line ports.

xPC Target supports this board with these driver blocks:

- Burr-Brown PCI-20098C Analog Input (A/D)
- Burr-Brown PCI-20098C Digital Input
- Burr-Brown PCI-20098C Digital Output

<b>Board Characteristics</b>	Board Name	PCI-20098C
	Manufacturer	Burr-Brown
	Bus Type	ISA
	Access Method	Memory mapped
	Multiple block instance support	A/D: No, Digital I/O: Yes
	Multiple board support	Yes



# Burr-Brown PCI-20098C Analog Input (A/D)

**Purpose** PCI-20098C Analog Input block

**Library** xPC Target Library for Burr-Brown

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

## **Block Parameters**

### **Number of Channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16 to select the number of single A/D channels used. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8. This driver does not allow the selection of individual channels or a different **Input coupling** setting for each channel.

### **Range**

From the list, choose either +-10V (-10 to +10 volts), +-5V (-5 to +5 volts), or 0-10V. This driver does not allow the selection of a different range for each channel. The input range is the same for all A/D channels

### **Input coupling**

From the list, select one from the following list of input modes:

- 16 single-ended channels
- 8 differential channels

This entry must correspond to the MUX-switch setting on the board.

### **Sample time**

Enter the base sample time or a multiple of the base sample time.

# Burr-Brown PCI-20098C Analog Input (A/D)

---

## **Base Address or Carrier Board (ie: 0xd000)**

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

# Burr-Brown PCI-20098C Digital Input

**Purpose** PCI-20098C Digital Input block

**Library** xPC Target Library for Burr-Brown

**Note** Use a separate driver block for each port. By selecting the digital input driver block, the port is configured as input.

## Scaling Input to Output

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

## Block Parameters

### Number of Channels

Enter a number between 1 and 8 to select the number of digital input lines used with this port.

### Port Number

From the list, choose either A or B to identify the port used with this block of I/O lines.

### Sample Time

Enter a base sample time or a multiple of the base sample time.

### Base Address or Carrier Board (ie: 0xd000)

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

# Burr-Brown PCI-20098C Digital Output

---

**Purpose** PCI-20098C Digital Output block

**Library** xPC Target Library for Burr-Brown

**Note** Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## **Block Parameters**

### **Number of Channels**

Enter a number between 1 and 8 to select the number of digital output lines used with this port.

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

### **Port Number**

From the list, choose either A or B to identify the port used with this block of I/O lines.

### **Sample time**

Enter a base sample time or a multiple of the base sample time.

### **Base Address or Carrier Board (ie: 0xd000)**

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

## **Gespac**

This section describes the Gespac boards supported by xPC Target (<http://www.gespac.com>).

# Gespac GESADA-1

---

## Board

Gespac GESADA-1

## General Description

The GESADA-1 is an industrial I/O board with 16 single or 8 differential analog input (A/D) channels and 4 analog output (D/A) channels (10-bit). xPC Target supports this board with these driver blocks:

- Gespac GESADA-1 Analog Input (A/D)
- Gespac GESADA-1 Analog Output (D/A)

---

**Note** xPC Target does not support the external trigger and interrupt propagation on this board.

---

## Board Characteristics

Board name	GESADA-1
Manufacturer	Gespac
Bus type	ISA industrial
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

# Gespac GESADA-1 Analog Input (A/D)

**Purpose** GESADA-1 Analog Input block

**Library** xPC Target Library for Gespac

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

## **Block Parameters**

### **Number of channels**

If you select 16 channels (**Input coupling** parameter set to 16 single-ended channels) enter a number between 1 and 16. If you select eight channels (**Input coupling** parameter set to 8 differential channels) enter a number between 1 and 8.

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

### **Range vector**

From the list, choose either +-10V (-10 to +10 volts), +-5V (-5 to +5 volts), or 0-10V. This driver does not allow you to select a different range for each channel. The input range is the same for all A/D channels.

The input range setting must correspond to the settings of jumper J6 and J9 on the board.

### **Input coupling**

From the list, select one from the following list of input modes:

- Single-ended channels (16 channels)
- Differential channels (8 channels)

This choice must correspond to the MUX-switch setting on the board.

# Gespac GESADA-1 Analog Input (A/D)

---

The differential mode is only supported if the board is equipped with option 1A. The MUX setting must correspond to the settings of jumper J3 and J7 on the board.

## **Sample time**

Base sample time or a multiple of the base sample time.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300



# Gespac GESADA-1 Analog Output (D/A)

**Purpose** GESADA-1 Analog Output block

**Library** xPC Target Library for Gespac

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

## Block Parameters

### Channel vector

This parameter is a combined channel vector and range vector. The number of elements defines the number of D/A channels used.

Enter a range code for each of the channels used. This driver allows a different range for each D/A channel with a maximum of two channels.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 to + 10 volts and the second, third, and fourth channel are -5 to +5 volts, enter

`[-10, -5, -5, -5]`

The range settings must correspond to the jumper setting of J5 on the board.

### Sample time

Enter the base sample time or a multiple of the base sample time.

# Gespac GESADA-1 Analog Output (D/A)

---

## **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

The base address specifies the base address of the board and has to correspond to the Jumper setting (J12) on the board.

**Board** Gespac GESPIA-2A

**General Description**

The GESPIA-2A is an industrial I/O board with 32 digital I/O lines. The GESPIA-2A has two 6821 PIAs (0 and 1) from Motorola. Each PIA has two ports (A and B) with 8 digital lints which can be defined as input or output.

xPC Target supports this board with these driver blocks:

- Gespac GESPIA-2A Digital Input
- Gespac GESPIA-2A Digital Output

**Board Characteristics**

Board name	GESPIA-2A
Manufacturer	Gespac
Bus type	ISA industrial
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

# Gespac GESPIA-2A Digital Input

---

**Purpose** GESPIA-2A Digital Input block

**Library** xPC Target Library for Gespac

**Note** Use a separate driver block for each port. By selecting the digital input driver block, the port is configured as input.

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

## **Block Parameters**

### **Number of channels**

Enter a number between 1 and 8 to select the number of digital input lines used with this port.

### **Port Name**

From the list, choose either PIA0A, PIA0B, PIA1A or PIA1B to identify the port used with this block of I/O lines.

### **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

# Gespac GESPIA-2A Digital Output

**Purpose** GESPIA-2A Digital Output block

**Library** xPC Target Library for Gespac

**Note** Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### Number of channels

Enter a number between 1 and 8 to select the number of digital output lines used with this port.

### Port Name

From the list, choose either PIA0A, PIA0B, PIA1A or PIA1B to identify the port used with this block of I/O lines.

### 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

# Gespac GESPIA-2A Digital Output

---

## **Measurement Computing (Computer Boards)**

This section describes the Measurement Computing boards supported by xPC Target (<http://www.measurementcomputing.com>).

There are new versions of the following Measurement Computing Incremental Encoder drivers. Please use the new version of these drivers. This topic describes the obsolete drivers:

- Measurement Computing CIO-QUAD02 Incremental Encoder
- Measurement Computing CIO-QUAD04 Incremental Encoder
- Measurement Computing PCI-QUAD04 Incremental Encoder

To access the previous versions of these drivers, use the `xpcobsoletelib` command. The presence of the obsolete versions of these drivers is not guaranteed for future releases.

# Measurement Computing CIO-QUAD02 Incremental Encoder

---

**Purpose** CIO-QUAD02 Incremental Encoder

**Library** xPC Target Library for Measurement Computing

**Description** This driver block has three block outputs: **Angle**, **Turns**, and **Init**. You can use **Init** to determine when the block output values are valid. **Init** is first set to 0. When the encoder reached the first index, **Init** is set to 1. From then on, you can determine the exact position, direction, and velocity. **Init** remains 1 unless **Counter Reset by Index** is set to **First Only**, and the counter detects a rollover. For more information, see “Block Parameters” on page 41-31.

**Turns** is the number of complete revolutions made by the encoder. **Angle** is the amount the encoder turns since the last full revolution.

The distance is given by:

$$\text{distance} = 2 * \pi * \text{Turns} + \text{Angle}$$

The velocity is given by:

$$\text{velocity} = (\text{distance}(t_s) - \text{distance}(t_{s-1})) / t_s$$

The direction is given by:

$$\text{direction} = \text{distance}(t_s) - \text{distance}(t_{s-1})$$

A negative value is reverse, while a positive value is forward.

## Block Parameters

### Function module

From the list choose 1 or 2. 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 Reset by Index

From the list choose either Only First, or Continuous.

If you choose Only First, the first time the encoder reaches the index, the counter is reset to zero, the **Init** output signal is set

# Measurement Computing CIO-QUAD02 Incremental Encoder

---

to 1, and the encoder begins to count. The encoder ignores all other times it reaches the index. **Init** remains 1 until a rollover is detected, and then set to -1. A rollover is when the counter reaches its maximum value and begins to start counting at zero again. A rollover can also occur when the counter reaches its minimum value and the counter resets itself to the maximum value and resumes counting down. The outputs are still accurate after rollover. The -1 flag is used to alert that a rollover has occurred.

If you choose Continuous, the first time the encoder reaches the index, the counter is reset to zero, the **Init** output signal is set to 1, and the encoder begins to count. Each time the encoder reaches the index, it resets to zero. **Init** remains always at 1 because a rollover cannot occur.

## Positive Rotation

From the list, choose either Clockwise or Counter Clockwise. This parameter sets the direction for positive rotation. If you choose Clockwise, when the encoder is turned clockwise it counts up, and when turned counter clockwise it counts down. If you choose Counter Clockwise the counting direction is reversed.

## Mode

From the list, choose Single, Double, or Quadruple. This parameter specifies the phase detection mode. That is, how many phase changes are detected. For more information, see the board manual.

## Resolution

This field specifies the divisions of the connected incremental encoder for one revolution.

## Filter prescale factor

Enter a base prescale factor for digital filtering. This filter helps eliminate high frequency noise. Enter a value from 0 to ff in hexadecimal (0 to 256 in decimal). For example, for a prescale factor value of ff in hexadecimal, enter

0xff



# Measurement Computing CIO-QUAD02 Incremental Encoder

---

## **Sample Time**

Enter a 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). Enter -1 to inherit the model sample time from another block. The sample time is in seconds.

## **Base Address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing CIO-QUAD04 Incremental Encoder

---

**Purpose** CIO-QUAD04 Incremental Encoder

**Library** xPC Target Library for Measurement Computing

**Description** This driver block has three block outputs: **Angle**, **Turns**, and **Init**. You can use **Init** to determine when the block output values are valid. **Init** is first set to 0. When the encoder reached the first index, **Init** is set to 1. From then on, you can determine the exact position, direction, and velocity. **Init** remains 1 unless **Counter Reset by Index** is set to **First Only**, and the counter detects a rollover. For more information, see “Block Parameters” on page 41-31.

**Turns** is the number of complete revolutions made by the encoder. **Angle** is the amount the encoder turns since the last full revolution.

The distance is given by:

$$\text{distance} = 2 * \pi * \text{Turns} + \text{Angle}$$

The velocity is given by:

$$\text{velocity} = (\text{distance}(t_s) - \text{distance}(t_s-1)) / t_s$$

The direction is given by:

$$\text{direction} = \text{distance}(t_s) - \text{distance}(t_s-1)$$

A negative value is reverse, while a positive value is forward.

## Block Parameters

### Function module

From the list choose, 1, 2, 3, or 4. 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 Reset by Index

From the list choose either Only First, or Continuous.

If you choose Only First, the first time the encoder reaches the index, the counter is reset to zero, the **Init** output signal is set

# Measurement Computing CIO-QUAD04 Incremental Encoder

---

to 1, and the encoder begins to count. The encoder ignores all other times it reaches the index. **Init** remains 1 until a rollover is detected, and then set to -1. A rollover is when the counter reaches its maximum value and begins to start counting at zero again. A rollover can also occur when the counter reaches its minimum value and the counter resets itself to the maximum value and resumes counting down. The outputs are still accurate after rollover. The -1 flag is used to alert that a rollover has occurred.

If you choose Continuous, The first time the encoder reaches the index, the counter is reset to zero, the **Init** output signal is set to 1, and the encoder begins to count. Each time the encoder reaches the index, it resets to zero. **Init** remains always at 1 because a rollover cannot occur.

## **Positive Rotation**

From the list, choose either Clockwise or Counter Clockwise. This parameter sets the direction for positive rotation. If you choose Clockwise, when the encoder is turned clockwise it counts up, and when turned counter clockwise it counts down. If you choose Counter Clockwise the counting direction is reversed.

## **Mode**

From the list, choose Single, Double, or Quadruple. This parameter specifies the phase detection mode. That is, how many phase changes are detected. For more information, see the board manual.

## **Resolution**

This field specifies the divisions of the connected incremental encoder for one revolution.

## **Filter prescale factor**

Enter a base prescale factor for digital filtering. This filter helps eliminate high frequency noise. Enter a value from 0 to ff in hexadecimal (0 to 256 in decimal). For example, for a prescale factor value of ff in hexadecimal, enter

0xff

# Measurement Computing CIO-QUAD04 Incremental Encoder

---

## **Sample time**

Enter a 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). Enter -1 to inherit the model sample time from another block. The sample time is in seconds.

## **Base address**

Enter the base address of the board. It is important that this entry corresponds to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Measurement Computing PCI-QUAD04 Incremental Encoder

---

**Purpose** PCI-QUAD04 Incremental Encoder block

**Library** xPC Target Library for Measurement Computing

**Description** This driver block has three block outputs: **Angle**, **Turns**, and **Init**. You can use **Init** to determine when the block output values are valid. **Init** is first set to 0. When the encoder reached the first index, **Init** is set to 1. From then on, you can determine the exact position, direction, and velocity. **Init** remains 1 unless **Counting reset by index** is set to Only First, and the counter detects a rollover. For more information, see “Block Parameters” on page 41-31.

**Turns** is the number of complete revolutions made by the encoder. **Angle** is the amount the encoder turns since the last full revolution.

The distance is given by:

$$\text{distance} = 2 * \pi * \text{Turns} + \text{Angle}$$

The velocity is given by:

$$\text{velocity} = (\text{distance}(t_s) - \text{distance}(t_s-1)) / t_s$$

The direction is given by:

$$\text{direction} = \text{distance}(t_s) - \text{distance}(t_s-1)$$

A negative value is reverse, while a positive value is forward.

## Block Parameters

### Function module

From the list choose 1, 2, 3, or 4. 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.

### Counting reset by index

From the list choose Only First, Continuous, or Index input disabled.

# Measurement Computing PCI-QUAD04 Incremental Encoder

---

If you choose `Only First`, the first time the encoder reaches the index, the counter is reset to zero, the **Init** output signal is set to 1, and the encoder begins to count. The encoder ignores all other times it reaches the index. **Init** remains 1 until a rollover is detected, and then set to -1. A rollover is when the counter reaches its maximum value and begins to start counting at zero again. A rollover can also occur when the counter reaches its minimum value and the counter resets itself to the maximum value and resumes counting down. The outputs are still accurate after rollover. The -1 flag is used to alert that a rollover has occurred.

If you choose `Continuous`, The first time the encoder reaches the index, the counter is reset to zero, the **Init** output signal is set to 1, and the encoder begins to count. Each time the encoder reaches the index, it resets to zero. **Init** remains always at 1 because a rollover cannot occur.

## **Positive Rotation**

From the list, choose either `Clockwise` or `Counter Clockwise`. This parameter sets the direction for positive rotation. If you choose `Clockwise`, when the encoder is turned clockwise it counts up, and when turned counter clockwise it counts down. If you choose `Counter Clockwise`, the counting direction is reversed.

## **Mode**

From the list, choose `Single`, `Double`, or `Quadruple`. This parameter specifies the phase detection mode. That is, how many phase changes are detected. For more information, see the board manual.

## **Resolution**

This field specifies the divisions of the connected incremental encoder for one revolution.

# Measurement Computing PCI-QUAD04 Incremental Encoder

---

## **Filter prescale factor**

Enter a base prescale factor for digital filtering. This filter helps eliminate high frequency noise. Enter a value from 0 to ff in hexadecimal (0 to 256 in decimal). For example, for a prescale factor value of ff in hexadecimal, enter

0xff

## **Sample time**

Enter a 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). Enter -1 to inherit the model sample time from another block. The sample time is in seconds.

## **PCI Slot (-1:autosearch)**

If only one board of this type is in the target PC, enter

-1

to automatically locate the board.